(1)

AFIT/GSO/ENS/92D-02

AN ANALYSIS OF USSPACECOM'S SPACE SURVEILLANCE NETWORK (SSN) SENSOR TASKING METHODOLOGY

THESIS

Jeff Mark Berger, Captain, USAF
Joseph Bruce Moles, Captain, USA
David George Wiisey, Captain, USAF

AFIT/GSO/ENS/92D-02

The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U. S. Government

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

AFIT/GSO/ENS/92D-02

AN ANALYSIS OF USSPACECOM'S SPACE SURVEILLANCE NETWORK SENSOR TASKING METHODOLOGY

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Space Operations

Jeff Mark Berger, B.S.
Captain, USAF

Joseph Bruce Moles, B.S., M.S.
Captain, USA

David George Wilsey, B.S.
Captain, USAF

December 1992

THESIS APPROVAL

STUDENTS: Captain Jeff M. Berger, USAF
Captain Joseph B. Moles, USA
Captain David G. Wilsey, USAF

CLASS: GSO-92D

THESIS TITLE: An Analysis of USSPACECOM's Space Surveillance Network Sensor Tasking Methodology

DEFENSE DATE: November 24, 1992

COMMITTEE:	*NAME/DEPARTMENT*	*SIGNATURE*
Advisor:	Thomas S. Kelso, Lt Col, USAF Asst Professor of Space Operations Dept of Operational Sciences School of Engineering	Thomas Sean Kelso
Co-Advisor:	William E. Wiesel, Jr. Professor of Astronautical Engineering Dept of Aeronautics and Astronautics School of Engineering	William E. Wiesel Jr

Acknowledgments

In performing this research, we have received help from many sources. We would like to thank our research advisor, LtCol Thomas S. Kelso for his continued patience and exhaustive assistance throughout our work. We would also like to thank our co-advisor, Dr. William E. Wiesel, for his help in the technical aspects of our research. Thanks are also due to Capt Gregory Bishop of the 1st Command and Control Squadron for information on NORAD operations provided during numerous telephone and personal interviews. Finally, CPT Moles would like to thank his wife Mary for her patience and support during the months of this research effort.

Jeff Mark Berger
Joseph Bruce Moles
David George Wilsey

Table of Contents

		Page
Appendix B.	SGP LIBRARY SOURCE LISTINGS	B-1
B.1	SGP4SDP4 Unit Source Code Listing	B-1
B.2	SGP_INIT Unit Source Code Listing	B-14
B.3	SGP_INTF Unit Source Code Listing	B-16
B.4	SGP_MATH Unit Source Code Listing	B-17
B.5	SGP_TIME Unit Source Code Listing	B-19
B.6	SUPPORT Unit Source Code Listing	B-22
B.7	SGP_CONV Unit Source Code Listing	B-28
B.8	SGP_IN Unit Source Code Listing	B-29
B.9	SGP_OUT Unit Source Code Listing	B-35
B.10	SGP_OBS Unit Source Code Listing	B-38
B.11	SOLAR Unit Source Code Listing	B-41
B.12	MINMAX Unit Source Code Listing	B-43
Appendix C.	TRUTH MODEL	C-1
C.1	HPOP_IN Source Code Listing	C-1
C.2	HPOP_IN Output/HPOP Input File Format	C-3
C.3	Sample HPOP_IN Output/HPOP Input File	C-3
C.4	Sample HPOP Output/CONVERT Input File	C-3
C.5	CONVERT Program Source Code Listing	C-4
C.6	RSELECT Program Source Code Listing	C-7
C.7	GAUSS2 Unit Source Code Listing	C-10
C.8	Sample RSELECT Output/DIFC Input File	C-11
Appendix D.	DIFFERENTIAL CORRECTOR PROGRAM	D-1
D.1	DIFC Program Source Code Listing	D-1
D.2	DC_INIT Unit Source Code Listing	D-3
D.3	DC_CALC Unit Source Code Listing	D-4

Page

List of Figures

Figure | Page

Figure		Page

Figure		Page

Figure		Page
H.24.	Class: 1–3–2 (Catalog Number 14199), LUPI 6, OPD 6 and 8. . . .	H-27
H.25.	Class: 1–3–2 (Catalog Number 14199), LUPI 8, OPD 2 and 4. . . .	H-28
H.26.	Class: 1–3–2 (Catalog Number 14199), LUPI 8, OPD 6 and 8. . . .	H-29
H.27.	Last-Pass — Class: 1–3–2 (Catalog Number 14199), LUPI 8, OPD 8.	H-30
H.28.	Class: 2–1–3 (Catalog Number 10293), LUPI 2, OPD 2 and 4. . . .	H-32
H.29.	Class: 2–1–3 (Catalog Number 10293), LUPI 2, OPD 6 and 8. . . .	H-33
H.30.	Class: 2–1–3 (Catalog Number 10293), LUPI 4, OPD 2 and 4. . . .	H-34
H.31.	Class: 2–1–3 (Catalog Number 10293), LUPI 4, OPD 6 and 8. . . .	H-35
H.32.	Class: 2–1–3 (Catalog Number 10293), LUPI 6, OPD 2 and 4. . . .	H-36
H.33.	Class: 2–1–3 (Catalog Number 10293), LUPI 6, OPD 6 and 8. . . .	H-37
H.34.	Class: 2–1–3 (Catalog Number 10293), LUPI 8, OPD 2 and 4. . . .	H-38
H.35.	Class: 2–1–3 (Catalog Number 10293), LUPI 8, OPD 6 and 8. . . .	H-39
H.36.	Last-Pass — Class: 2–1–3 (Catalog Number 10293), LUPI 8, OPD 8.	H-40
H.37.	Class: 2–1–4 (Catalog Number 10393), LUPI 2, OPD 2 and 4. . . .	H-42
H.38.	Class: 2–1–4 (Catalog Number 10393), LUPI 2, OPD 6 and 8. . . .	H-43
H.39.	Class: 2–1–4 (Catalog Number 10393), LUPI 4, OPD 2 and 4. . . .	H-44
H.40.	Class: 2–1–4 (Catalog Number 10393), LUPI 4, OPD 6 and 8. . . .	H-45
H.41.	Class: 2–1–4 (Catalog Number 10393), LUPI 6, OPD 2 and 4. . . .	H-46
H.42.	Class: 2–1–4 (Catalog Number 10393), LUPI 6, OPD 6 and 8. . . .	H-47
H.43.	Class: 2–1–4 (Catalog Number 10393), LUPI 8, OPD 2 and 4. . . .	H-48
H.44.	Class: 2–1–4 (Catalog Number 10393), LUPI 8, OPD 6 and 8. . . .	H-49
H.45.	Last-Pass — Class: 2–1–4 (Catalog Number 10393), LUPI 8, OPD 8.	H-50
H.46.	Class: 2–2–3 (Catalog Number 19859), LUPI 2, OPD 2 and 4. . . .	H-52
H.47.	Class: 2–2–3 (Catalog Number 19859), LUPI 2, OPD 6 and 8. . . .	H-53
H.48.	Class: 2–2–3 (Catalog Number 19859), LUPI 4, OPD 2 and 4. . . .	H-54
H.49.	Class: 2–2–3 (Catalog Number 19859), LUPI 4, OPD 6 and 8. . . .	H-55
H.50.	Class: 2–2–3 (Catalog Number 19859), LUPI 6, OPD 2 and 4. . . .	H-56

Figure Page

List of Tables

AFIT/GSO/ENS/92D-02

Abstract

This study provides the basis for the development of a cost/benefit assessment model to determine the effects of alterations to the Space Surveillance Network (SSN) on orbital element (OE) set accuracy. It provides a review of current methods used by NORAD and the SSN to gather and process observations, an alternative to the current Gabbard classification method, and the development of a model to determine the effects of observation rate and correction interval on OE set accuracy. The proposed classification scheme is based on satellite J_2 perturbations. Specifically, classes were established based on mean motion, eccentricity, and inclination since J_2 perturbation effects are functions of only these elements. Model development began by creating representative sensor observations using a highly accurate orbital propagation model. These observations were compared to predicted observations generated using the NORAD Simplified General Perturbation (SGP4) model and differentially corrected using a Bayes, sequential estimation, algorithm. A 10-run Monte Carlo analysis was performed using this model on 12 satellites using 16 different observation rate/correction interval combinations. An ANOVA and confidence interval analysis of the results show that this model does demonstrate the differences in steady state position error based on varying observation rate and correction interval.

AN ANALYSIS OF USSPACECOM'S SPACE SURVEILLANCE NETWORK SENSOR TASKING METHODOLOGY

I. INTRODUCTION

1.1 Background

Following the launch of Sputnik I on 4 October 1957, the United States realized that it needed the capability to track and identify objects in earth orbit (7:12-10). Shortly thereafter, development began on a system of sensors to perform that mission. Today, it is the responsibility of the United States Space Command (USSPACECOM) Space Surveillance Center (SSC), at Cheyenne Mountain Air Force Base (CMAFB) in Colorado, to detect, track, identify, and catalog all objects in earth orbit (26:1) (23:39). To perform these tasks, the SSC requires accurate positional data on all of the objects in orbit. This data is provided by the 25 worldwide sensors of the Space Surveillance Network (SSN).

The sensors of the SSN continuously track approximately 6,000 man-made objects in earth orbit and gather metric (positional) data on them. The sensors send these observations (approximately 40,000 observations a day) to the SSC. The SSC uses this data to classify and identify all detected objects and maintain the orbital element (OE) sets on each of these objects (18:8). The OE sets are used to predict the position of any object at any time. These predictions have many important operational applications that include: collision avoidance, satellite decay and impact predictions, satellite maneuver identification, warning of a satellite passing over a specific geographic area, and warning of attack on US space assets (7:12-10, 12-11). Therefore, maintaining the accuracy of the OE sets used to make these predictions is essential.

Based on the current satellite growth rate, the SSC will be maintaining surveillance on about 10,000 objects by the year 2000 (17:588). Correspondingly, the number of observations on these objects received and processed by the SSC will also rise. Given this growth rate, the tracking limits of the SSN, and current computational limits, the capacity

of the SSC will eventually be reached. In order to maintain the required orbit prediction accuracy on this steadily increasing number of objects, new or upgraded sensors and computers will be required. However, in the future military budget climate, money for these upgrades may not be available. USSPACECOM may also find itself under pressure to shut down some of its older sensors that are deemed too expensive to operate.

1.2 Problem

The problem which USSPACECOM will eventually be faced with is how to maintain orbit prediction and OE set accuracy on an increasing number of objects while possibly having to (1) decrease the number of sensors, (2) decide where to place new sensors, and/or (3) decide which sensor(s) or computer system(s) to upgrade with limited available funding. In other words, how to operate the SSC and SSN as efficiently as possible with minimum resources (sensors and computer systems).

1.3 Research Objective

There are two objectives of this research. The first is to analyze USSPACECOM's current sensor tasking methodology, focusing on current operational procedures and methods of classifying the earth satellite population. The second is to develop and demonstrate a model from which you can determine the effects of observation rate and correction interval on the accuracy of an OE set. This research is intended to provide the basis for the development of a quantitative cost/benefit assessment model to allow USSPACECOM to assess the effects of alterations to the SSN on orbit prediction accuracies.

This research is being performed at the request of the 1st Command and Control Squadron at Cheyenne Mountain AFB, Colorado.

1.4 Scope of Research

This effort is divided into five specific areas of research and analysis. The first area will be an analysis of the SSN. The second area will be an analysis of the earth satellite populaton and USSPACECOM's current method of classifying satellites. The third area

will be the development of a highly accurate "truth" model used to produce simulated observations from the sensors of the SSN. The fourth area will be the development of a differential corrector (DC) used to correct satellite orbital elements based on a set of "truth" observations. The last area will be a statistical analysis of output from the differential corrector to estimate the position accuracy of various observation rate - correction interval combinations on a set of representative satellites.

1.4.1 Analysis of the SSN. The purpose of this analysis is to determine the methods currently used to task sensors of the SSN to gather observational data, and to update satellite orbital element sets. This data will establish a baseline for comparison with results from the dynamics model. In the performance of this analysis, the following specific questions will be examined:

- What are the locations and operational characteristics of SSN sensors?
- What are the capabilities of the SSN?
- How are the sensors of the SSN tasked and how was that method developed?
- How are the sensor observation data processed?
- What are the operational requirements for orbital element accuracy?
- How does the SSN classify the objects it tracks?
- How does USSPACECOM update orbital elements?
- How does atmospheric refraction affect SSN sensor observations?
- What are the error characteristics of the SSN sensors?

The answers to these questions, and the results of this analysis, will be presented in the Literature Review.

1.4.2 Analysis of Earth Satellite Population. In addition to understanding the current methods used to task sensors to gather observational data on earth satellites and to update satellite orbital elements, an understanding of the satellite orbital characteristics is essential. The purpose of analyzing the earth satellite population is to examine the orbital characteristics of satellites in earth orbit and select a representative sample for analysis.

1.4.3 Truth Model Development. The truth model is intended to produce highly accurate "truth" observations of earth orbiting satellites. These truth observations will simulate actual SSN sensor observations received by USSPACECOM and be used to correct OE sets.

1.4.4 Differential Corrector Development. The differential corrector model is intended to update satellite orbital elements based on simulated observations from the truth model and output the vector magnitude of the error between the "truth" position and the predicted, position for each observation. Since the results from this model will be compared to actual data from USSPACECOM, we will attempt to simulate methods used by USSPACECOM as closely as possible and model the composition and characteristics of the SSN as realistically as possible.

1.4.5 Statistical Analysis. A statistical analysis of the output from the differential corrector will be performed to determine the statistical mean and variance of the steady-state position error of a satellite based on a given observation rate and correction interval. These estimations can, in turn, be used to show the effect of observation rate and correction interval on the true position error of a satellite.

1.5 Limitations of Research

In performing this research, we will limit our analysis to only objects cataloged by NORAD as of 1 March 1990. Due to various problems encountered in transferring orbital element set data from USSPACECOM to AFIT, and time restrictions due to thesis submittal requirements, this set was the most current complete set of data available for analysis.

II. LITERATURE REVIEW

2.1 Introduction

This purpose of this chapter is to provide a review of literature and information pertinent to this research. The goal of the literature review is to lay the ground work for the research effort. The following subjects are covered in the review:

- SSN sensor types and characteristics.
- Composition of the SSN.
- SSN sensor locations, limits, and coverage areas.
- Error associated with SSN sensors.
- SSC satellite classification methodology.
- SSC sensor tasking methodology.
- SSN sensors observation processing methodology.
- SSC observation processing and orbital element correction methodology.
- Previous analyses of SSN capabilities.

2.2 SSN Sensor Types and Characteristics

The SSN uses two basic types of sensors: optical and radar sensors. Optical sensors measure the visible energy emitted or reflected by objects and are typically used to track deep-space objects.[1] Radar sensors measure high-frequency radio waves reflected by objects and are typically used to track both near-earth[2] and deep-space objects.

2.2.1 Radar Sensors. The development of radar has been called the greatest advance in the remote sensing of objects since the invention of the telescope in 1608 (1:1). The radar saw its first major use during World War II and the Battle of Britain. Since

[1]Orbital period greater than or equal to 225 minutes (24:223).
[2]Orbital period less than 225 minutes (24:223).

that time, the basic principle of the system has not changed but the technology supporting the system has changed dramatically.

Regardless of the type of radar, all radars operate on the same general principle. A beam of electromagnetic energy is transmitted from an antenna toward a target. The physical characteristics of the target will determine if the energy is absorbed, reflected, or some combination of both. If any of the energy is reflected, the target itself becomes an antenna and re-emits the electromagnetic energy in all directions. The direction of the strongest reflection is typically in the direction of the radar transmitter.

In general, for a given observation, the total reflected energy received by the radar may be used to characterize the target in terms of detectability and measuring ability of the radar. The ability of a radar to detect and track a target can be shown to be a function of the average power of the transmitter, the time the target is illuminated by the radar beam, and the geometry of the radar-target situation (1:4).

An important defined quantity based on the geometry of a target is called radar cross section (RCS). RCS (σ) is defined as

$$\sigma = 4\pi \times \frac{\Psi_r}{\Psi_t} \tag{2.1}$$

where Ψ_r is defined as the reflected power per unit solid angle in the direction of the source, and Ψ_t is defined as the power per unit area of the transmitted signal (1:66, 110). The units for σ are generally m^2 or dBm^2. If a target were to scatter the radiation uniformly, the RCS would be the area from which the power was extracted from the incident wave (1:66). If the target can be modeled as a sphere with constant area cross section from all viewing angles, the RCS may be modeled as a single value (1:80). The larger the value for RCS, the larger the quantity of reflected energy or target echo.

The echo returned by a simple point target will be an exact reproduction of the transmitted signal. However, the signal is shifted in time due to range delay, shifted in frequency by the Doppler shift due to the target's radial velocity, and shifted in amplitude due to the geometry of the radar and target situation (1:2-3). By analyzing these shifts in frequency, the range and velocity of the target may be determined.

2.2.1.1 Radar Functions. There are two basic functions carried out by a radar: search and detection (29:4). Search is the methodology used by the radar to look for targets within a specified field of view. Detection is the ability of the radar to identify a target in its field of view. These terms are independent of the type of radar being discussed. However, the method of employment varies for different types of radars.

For mechanical radars, the search methodology is based on a rotating antenna or rotating feedhorn. The assembly may be sending energy, receiving energy, or both. Another name for the rotating antenna is *mechanical tracker* and is the name most often used in the SSN. If the same antenna transmits and receives, the system is called a *mono-static* system. However, if separate antennas transmit and receive, the system is referred to as a *bi-static* system.

For a phased-array radar, the searching methodology is based on electronically changing the phase of the transmitted signal. The face of the radar is composed of groupings of electronically steerable antennas. The individual antennas emit phase-coherent radiation to emit a plane wave of radiation in the desired direction. However, since the phased-array radar is not limited to a direction normal to the antenna as the mechanical radar is, the search methodology varies greatly from the methodology of the mechanical tracker (29:8).

The search philosophy for each type of radar is different based on its strengths and limitations. However, both methods are based on the probability of a target being in a specific sector of observation. While a mechanical tracker may search in a circular pattern, the probability of detecting the target improves if the location of the target can be limited to a certain sector for search. In a similar manner, the phased-array develops an *a posteriori* probability of a target being in a certain cell of observed space. The search is then begun in the cell with the highest probability of containing a target (29:8).

Detection, the second function of a radar, is the process of determining the presence of a target in the presence of competing electromagnetic signals. These competing signals arise from many sources. The most common sources of these signals are background radiation, undesirable echo, or the radar receiver (1:1). The limiting precision (standard deviation) of angular measurements associated with target detection will be on the order of

one tenth of the half-power beamwidth for the targets which can be distinguished against their background radiation (1:54).

Assuming the radar search method locates and then detects the target, the electromagnetic energy may then be used to determine several characteristics of the target. The characteristics of interest are velocity, range, azimuth and elevation.

The time delay between the transmitted pulse and the corresponding received pulse will be an indication of the target range. This relation ship may be expressed as:

$$\Delta t_r = \frac{2R}{c} \tag{2.2}$$

where c is the speed of light, Δt_r is the return delay time of the pulse, and R is the range (1:3).

The velocity of the target may be calculated from the shift of the center frequency of the radar pulse. The center frequency of the returned pulse will be shifted from the transmitted pulse, f_t, by:

$$f_t + f_d = f_t \frac{(c + v_r)}{(c - v_r)} \tag{2.3}$$

where f_d is the frequency of the Doppler shift, c is the velocity of light in a vacuum, and v_r is the radial velocity of the target (1:3).

The remaining characteristics are azimuth and elevation. With these additional pieces of information, the position of the target may be determined in detail. To determine azimuth and elevation, measurements are taken directly from the mechanical devices accomplishing the tracking (mechanical tracker) or from the electronics controlling the wave front (phased-array).

2.2.1.2 SSN Radar Sensors. There are three types of radar sensors in the SSN: mechanical radars, phased-array radars, and a radar interferometer. Mechanical radars are either fan type or steerable trackers. The fan-type radars are large fixed antennas with mechanically moving feeds which scan a swath of space a couple degrees above the

horizon. Primarily designed to detect incoming ballistic missiles, they can also detect orbiting objects as they pass through the radar's field of view (FOV) (26:6). Mechanical steerable tracking radars use dish-type antennas and can track only one object at a time. Most radars of this type are used to track deep-space objects.

Phased-array radars are the backbone of the SSN, providing over 60 percent of the observations sent to the SSC (17:585). They electronically steer the tracking beam by phasing the energy emitted from thousands of small transmitters. They are capable of directing the beam in many directions and switching targets in milliseconds. Thus, they can simultaneously track several hundred objects (17:585) (26:6).

The one radar interferometer in the SSN consists of three transmitters and six receivers, positioned roughly along the 33rd parallel between Georgia and California (23:40). Each transmitter and receiver is approximately 30 feet wide and 2 miles long. Together, they create, in effect, a radar fence 5,000 miles long and 15,000 miles high capable of detecting all satellites with an inclination greater than 33 degrees[3] (17:585).

2.2.2 Optical Sensors. An optical sensor must also search for and detect a satellite crossing the sky to provide data to update orbital elements. As the sensor tracks satellites across the sky, changes in right ascension and declination are recorded. These changes in angles can be converted into orbital elements for the target satellite. The accuracy of the sensor in measuring these angles directly affects the accuracy of the orbital elements.

The basic requirements for an electro-optical tracking device are: sensor, positioning system, and a command and control system for the positioning system (6:70). More specifically, the system must comprise a sensor, gimbal, tracker, and gimbal control.

The sensor is composed of an electronic detector, optics, and associated electronics. The Ground-based Electro-Optical Deep-Space Surveillance (GEODSS) sensors are the primary optical sensors used by the SSN. GEODSS uses a low-light-level charge-coupled-device (CCD), computer analysis, and large telescopes to provide observational data to the SSC in real time (7:12-12).

[3]Approximately 85 percent of the current satellite population. See Figure 5.5.

The gimbal is a mounting device which stabilizes the sensor and provides control of the tracking motion. The tracker is an additional piece of equipment which identifies the target and keeps the sensor pointing at the target. The CCD is an electronic device which records the impact of photons on the sensor. The gimbal control is the motor, gears and electronics required to operate the gimbal (6:72–73).

2.3 Space Surveillance Network Composition

The Space Surveillance Network is a conglomerate of 25 ground-based radar and optical sensors used for space surveillance (23:42–43). The 25 sensors are of 19 different designs, with only seven of the 25 sensors being designed specifically for the SSN as tracking sensors (18:7) (23:39–43). The remaining 19 sensors were originally designed with other primary missions, such as ballistic missile warning, missile testing, or scientific research and development, and were adapted to the SSN because of their capabilities (23:39–43).

Because of inherent limitations, most of the sensors of the SSN are unable to track objects smaller than 10 cm in diameter (17:587). In addition to the 7,000 objects larger than 10 cm currently tracked, it is estimated that there are 48,000 to one million objects ranging in size from one cm to 10 cm in diameter. It is believed that objects less than one cm number into the billions (17:587) (23:39).

The sensors of the SSN are operationally divided into three categories based on which agency has operational control over the sensor and when the sensors provide support to the SSN. Those categories are: dedicated, collateral, and contributing sensors.

2.3.1 Dedicated Sensors. Dedicated sensors are those with a primary mission of space surveillance. There are seven dedicated sensors in the SSN with the majority of them being optical systems. The premier dedicated system is the GEODSS. GEODSS was designed to replace the Baker-Nunn camera system (7:12-12). There are four operational GEODSS sites (with one more planned) roughly spaced evenly around the globe to give complete coverage of all satellites in geosynchronous orbit (17:586). The operational sites are: Socorro, New Mexico; Taegu, South Korea; Maui, Hawaii; and Diego Garcia in the Indian Ocean (23:39).

The Maui Optical Tracking Identification Facility (MOTIF) is co-located with the GEODSS site at Maui. It is similar in capability to the GEODSS systems but also has an infrared detection capability (23:39). Both of these optical systems are limited to night observations in clear weather (23:39).

The radar at Eglin AFB, Florida is the only dedicated phased-array radar sensor in the SSN. It is also the only phased-array radar capable of tracking both near-earth and deep-space objects (23:39). The radar interferometer, or Naval Space Surveillance (NAVSPASUR) System, operated by the Navy, is also a dedicated sensor.

2.3.2 Collateral Sensors. Collateral sensors are those sensors operationally controlled by USSPACECOM with a primary mission other than space surveillance. They support the SSN when not performing their primary missions. There are ten collateral sensors in the SSN. They consist primarily of the phased-array radars ringing the Northern Hemisphere, watching for ballistic missile launches.

The PAVE PAWS system consists of four identical two-face phased-array radars with a primary mission of SLBM detection. Space surveillance is their secondary mission (23:40). The radars are located at: Cape Cod AFB, Massachusetts; Beale AFB, California; Robins AFB, Georgia; and Eldorado AFB, Texas. These radars track only near-earth objects (18:7).

The Ballistic Missile Early Warning System (BMEWS) consists of a mechanical fan radar at Clear AFB, Alaska, a two-faced phased-array radars at Thule AFB, Greenland and a three-face phased-array radar at Fylingdales Moor, United Kingdom. These radars have a primary mission of ICBM detection and a secondary mission of space surveillance (23:40). These radars track only near-earth objects (18:7).

The Perimeter Acquisition Radar Attack Characterization System (PARCS) is a phased-array radar located at Cavalier, North Dakota. It has a primary mission of SLBM detection and a secondary mission of space surveillance (23:40). It tracks only near-earth objects (18:7).

The last two collateral sensors are the COBRA DANE phased-array radar at Shemya AFB, Alaska, and the steerable mechanical-tracking radar at Pirinclik, Turkey. Both have

a primary mission of intelligence collection and a secondary mission of space surveillance (23:39). They both track only near-earth objects (18:7).

2.3.3 Contributing Sensors. Contributing sensors are those sensors not under direct USSPACECOM operation control, but which, by agreement, provide support to the SSN when not performing their primary mission. There are eight contributing sensors in the SSN.

The Air Force Maui Optical Station (AMOS) is a optical telescope system which performs scientific research and development tasks for Air Force Material Command. It is co-located with the MOTIF and GEODSS systems on Maui (23:40). It is used for tracking deep-space objects (18:7).

The steerable mechanical-tracking radars on Kaena Point in Hawaii, and Antigua and Ascension Islands in the Atlantic Ocean are used to support missile testing out of the Western and Eastern test ranges (23:40). These sensors all perform near-earth tracking (18:7).

The Advanced Research Projects Agency (ARPA) Lincoln Tracking and Identification Radar (ALTAIR) and the ARPA-Lincoln Coherent Observables Radar (ALCOR) on Kwajalein Atoll in the Pacific Ocean are operated by the Army in support of Army, Navy, and Air Force missile testing (23:40). ALTAIR performs deep-space tracking while ALCOR, performs near-earth tracking (18:7).

Finally, the Millstone and Haystack radars located on Millstone Hill in Massachusetts perform scientific research and development. They are operated by Lincoln Labs for the Massachusetts Institute of Technology (23:40). When supporting the SSN, they both perform deep-space tracking (18:7).

Table 2.1 provides a summary of sensor category, sensor type (phased-array radar, mechanical-tracking radar, radar interferometer, or optical sensor), and sensor mission (near-earth or deep-space).

Table 2.1. SSN Sensor Category, Type and Mission.

Sensor Number(s)	Sensor Name	Sensor Category	Sensor Type	Sensor Mission
211, 212, 213	GEODSS SOCORRO	Dedicated	EO	DS
221, 222, 223	GEODSS TAEGU	Dedicated	EO	DS
231, 232, 233	GEODSS MAUI	Dedicated	EO	DS
241, 242, 243	GEODSS DIEGO GARCIA	Dedicated	EO	DS
333	ALCOR	Contributing	MTR	NE
334	ALTAIR	Contributing	MTR	DS
337,401,403,404	PIRINCLIK	Collateral	MTR	NE and DS
344, ???, ???	FYLINGDALES*	Collateral	PAR	NE
349, 359	CLEAR	Collateral	MTR	NE
354, 355	ASCENSION	Contributing	MTR	NE
363	ANTIGUA	Contributing	MTR	NE
369	MILLSTONE HILL	Contributing	MTR	DS
370	MILLSTONE UHF	Contributing	MTR	DS
382, 383	GOODFELLOW	Collateral	PAR	NE
384, 385	ROBINS	Collateral	PAR	NE
386, 387	CAPE COD	Collateral	PAR	NE
388, 389	BEALE	Collateral	PAR	NE
393	COBRA DANE	Collateral	PAR	NE
394, 395	THULE	Collateral	PAR	NE
396	PARCS	Collateral	PAR	NE
398, 399	EGLIN	Dedicated	PAR	NE and DS
741 – 747	NAVSPASUR	Dedicated	RIF	NE
932	KAENA POINT	Contributing	MTR	NE
951	MOTIF	Dedicated	O	DS
952	AMOS	Contributing	O	DS

* New three-face phased-array radar. Sensor numbers for two-faces unknown.

PAR = Phased-Array Radar
MTR = Mechanical-Tracking Radar
RIF = Radar Interferometer
O = Optical sensor
EO = Electro-Optical sensor

DS = Deep-Space satellite tracking
NE = Near-Earth satellite tracking

Compiled from (35:3-4), (36:16-18) and (11:4-2)

2.4 *Space Surveillance Network Coverage and Limitations*

To provide the best possible coverage of the space environment, USSPACECOM has attempted to locate the sensors around the earth based on mission requirements and limitations of the sensor. All collateral and contributing sensors were located in the areas best suited to support their primary missions. The GEODSS optical deep-space sensors were located as near the equator and as equally spaced about the earth as possible to provide optimal viewing geometry for all geostationary satellites.

Table 2.2 contains a summary of the various sensor locations. Table 2.3 contains a summary of the range, azimuth, and elevation limitations for the sensors. Figure 2.1 is a representation of the worldwide locations of SSN sensors.

2.5 *Space Surveillance Network Sensor Accuracies*

2.5.1 Radar Sensor Error. Error is the difference in the true position of the target and the position indicated by the radar. The purpose of error analysis is to provide a description of this error. This description will allow the magnitude of error to be estimated under any set of operating conditions. In general, the error will be a function of the time of measurement, the value of the quantity to be measured, and the environmental conditions present during the measurement (1:318).

For purposes of analysis, error is commonly divided into two components: systematic (bias) and random (noise). Systematic errors are characterized by their predictability and may be eliminated from the final measurement. If the error is the same for all possible observation conditions, a single number may be used to represent the error and then subtracted from the measurement. In general, the measured values will vary around a mean value (true bias) (1:318).

If the bias error is constant for extended lengths of time compared to the calibration and operation time of the system, then the error may be removed through calibration. If errors are introduced as a function of the measured quantity, then the speed at which this quantity varies will determine how much of the error appears as systematic and random error.

Table 2.2. SSN Sensor Locations.

Sensor Number(s)	Sensor Name	North Latitude	East Longitude	Altitude
		degrees	degrees	meters
211, 212, 213	GEODSS SOCORRO*	33.817	253.340	1508.5
221, 222, 223	GEODSS TAEGU*	35.744	128.608	782.2
231, 232, 233	GEODSS MAUI*	20.708	203.742	3056.5
241, 242, 243	GEODSS DIEGO GARCIA*	-7.411	72.452	-63.0
333	ALCOR	9.395390	167.479131	60.7
334	ALTAIR	9.395390	167.479131	60.7
337,401,403,404	PIRINCLIK†	37.905219	39.993182	887.9
344, ???, ???	FYLINGDALES‡	54.37	359.33	298.0
349, 359	CLEAR†	64.291157	210.807021	211.5
354. 355	ASCENSION*	-7.940	345.598	97.0
363	ANTIGUA	17.143601	298.207327	-1.7
369	MILLSTONE HILL	42.617404	288.508954	121.0
370	MILLSTONE UHF	42.619566	288.508606	111.1
382, 383	GOODFELLOW§	30.978253	259.447024	772.1
384, 385	ROBINS§	32.581227	276.430640	83.9
386, 387	CAPE COD§	41.752423	289.461731	78.6
388, 389	BEALE§	39.136044	238.649121	113.9
393	COBRA DANE	52.737262	174.090931	89.4
394, 395	THULE§	76.570286	291.700759	422.6
396	PARCS	48.724788	262.100257	345.5
398, 399	EGLIN§	30.572425	273.785153	33.3
741 747	NAVSPASUR¶	33.0	Various	Various
932	KAENA-POINT	21.572087	201.733258	297.8
951	MOTIF	20.708462	203.742024	3057.4
952	AMOS	20.708311	203.742486	3056.2

* Average location and altitude for near co-located sensors.
† Multiple co located mechanical-tracking sensors.
‡ Actual position data for new three-face phased-array radar not available. Estimated position based on position data of old mechanical-tracking radars (341, 342, and 343).
§ Two-face phased-array radar with faces co-located.
¶ Consists of six sensors positioned roughly at 33 north latitude and evenly spaced between 243 and 276 east longitude.

Compiled from (35:3 12 to 3-28), (11:3-27), and (14)

Table 2.3. Space Surveillance Network Sensor Limits.

Sensor Number(s)	Sensor Name	Range Min	Range Max	Elevation Min	Elevation Max	Azimuth Min	Azimuth Max
		km	km	deg	deg	deg	deg
211, 212, 213	GEODSS SOCORRO	5555	40000	20	90	0	360
221, 222, 223	GEODSS TAEGU	5555	40000	20	90	0	360
231, 232, 233	GEODSS MAUI	5555	40000	20	90	0	360
241, 242, 243	GEODSS DIEGO GARCIA	5555	40000	20	90	0	360
333	ALCOR	0	4500	0	95.8	0	360
334	ALTAIR	5555	40000	2	92	0	360
337, 401	PIRINCLIK	0	5100	2	86	0	360
403, 404	PIRINCLIK	5555	40000	2	86	0	360
344, ???, ???	FYLINGDALES†	0	5555	3	85	0	360
349, 359	CLEAR	0	4910	1.5	90	0	360
354, 355	ASCENSION	0	1900	1	90	0	360
363	ANTIGUA	0	2550	0	90	0	360
369	MILLSTONE HILL	5555	40744	1	90	0	360
370	MILLSTONE UHF	5555	40744	0	85	0	360
382, 383	GOODFELLOW	0	5555	3	85	70	310
384, 385	ROBINS	0	5555	3	85	10	230
386, 387	CAPE COD	0	5555	3	85	347	227
388, 389	BEALE	0	5555	3	85	126	6
393	COBRA DANE	0	4500	0.6	80	259	19
394, 395	THULE	0	5555	3	85	297	177
396	PARCS	0	3300	1.9	95	298	78
398	EGLIN	5555	40000	1.9	105	120	240
399	EGLIN	0	13210	1.9	105	120	240
932	KAENA POINT	0	6380	0	90	0	360
951	MOTIF*	5555	40000	20	90	0	360
952	AMOS*	5555	40000	20	90	0	360

NOTE: For this research, all deep-space sensors assumed to have range 5555 – 40000 km, unless otherwise specified.

* No data available. Limits assumed same as GEODSS.

† Estimated limits for new three-face phased-array radar based on other PAVE PAWS radar limits.

Compiled from (35:3-12 to 3-28) and (14)

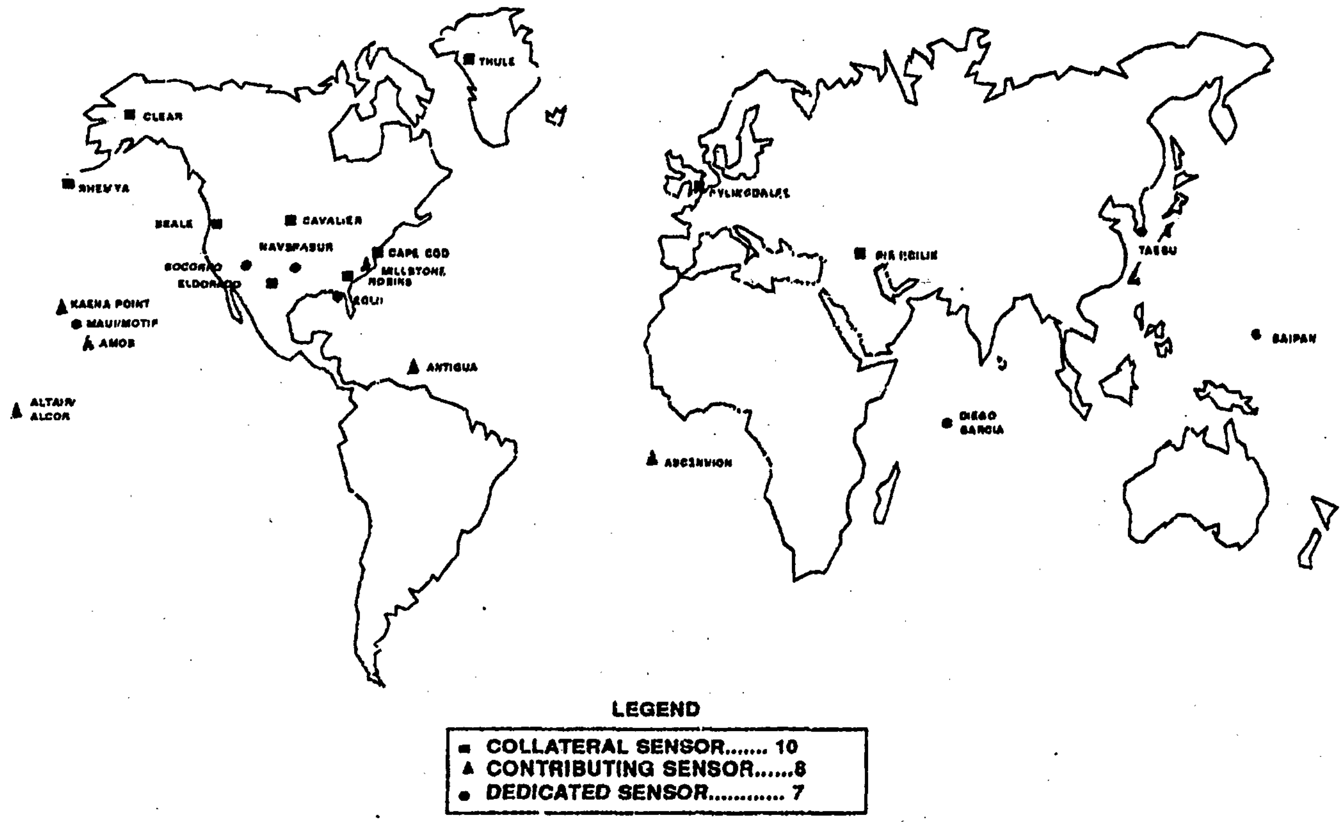


Figure 2.1. Worldwide SSN Sensor Locations.

This definition of systematic and random error implies there is no rigid dividing line between the two quantities. To break up the quantities, an arbitrary time must be chosen where a majority of error remains constant. The constant errors may then be classified as bias (systematic) and the rest accounted for as noise (random) (1:322).

A statistical description of errors allows the decomposition of the radar system into several independent components or elements. The components of error are then measured or calculated in a root-mean-square (RMS) process and added together to obtain an overall system error. The underlying assumption is the absence of correlation between the multiple elements. In practice, it has been found safe to ignore the correlation between multiple elements unless there is a clear physical link which would result in a common variation of two or more elements (1:324).

2.5.1.1 Noise. The principal source of noise in a radar system is thermal radiation. This radiation is received from the environment and is characterized as either internal or external to the system. External noise is defined as natural background radiation while internal noise is the interference created by the electrical components of the radar system. Additionally, noise may be created from the clutter of unwanted targets picked up by system side-lobe patterns (1:4-5).

As radars advanced, they obtained extended range and accuracy. As part of the development effort, several studies were performed to determine the effect of thermal noise. Using a Gaussian approximation of a pulsed beam shape, Swerling of RAND Corporation derived the following expressions for the limiting precision (standard deviation) on a constant amplitude target. Equation (2.4) is for (S/N) $\ll$ 1 and Equation (2.5) is for (S/N) $\gg$ 1.

$$\sigma_{min} = .58\frac{\Theta}{(S/N)\sqrt{n}} \tag{2.4}$$

$$\sigma_{min} = .49\frac{\Theta}{\sqrt{(S/N)n}} \tag{2.5}$$

where Θ is the half-power beamwidth, n is the number of pulses received between the half-power points on the one way pattern, and σ_{min} is the standard deviation (1:51).

2.5.1.2 Classification and Description of Errors. The list below characterizes sources of error in radar measurements. These errors may be seen in angle error, range error, or both. In general, the error is divided into electrical components, mechanical components, and environmental components.

- Radar-Dependent Tracking Errors: The bias components of the errors are mostly the result of boresight error, wind torque (in unshielded antennas), and servo unbalance and drift in moving antennas. The random component of this error comes from thermal noise and multipath effects, as well as the random portions of the bias elements (1:325–326).
- Multipath Error: The multipath effect comes from the reflection of the main lobe by the ground at low elevation angles. This error is reduced once the target rises more than one beamwidth above the horizon due to the relatively small beamwidths used for tracking applications. However, even very small signals will result in disturbing the null position of the tracking servo (1:327). Multipath is a major source of error in elevation measurements. The effect on azimuth is not as pronounced, but the effect on elevation is serious. Multiple reflections are most adverse when measurements are being taken as the target is near the horizon (1:55).
- Antenna/Servo-Torque Error: The analytical evaluation of this error requires extensive knowledge of the open-loop transfer function. The wind torque becomes greater as the size of an unshielded antenna increases (1:331–335).
- Collimation and Drift Errors: If the electrical system of the radar is stable, this error depends on the care exercised in calibration. Certain environmental factors contribute to this error and change too quickly to be removed by calibration. To conduct a complete review, the position of the axis must be determined as a function of: frequency of operation in the radar band, tuning of the system, phase or gain variations in the receivers, signal strength, and temperature or intensity of thermal radiation (1:335–339).

- Radar-Dependent Translation Errors: This refers to the translation of data into a usable form (1:339). These errors are most often the result of physical problems with the radar equipment.

- Non-Radar Components: These errors are mostly target dependent. They indicate the dependence of detection on the physical characteristics of the actual target. Additionally, atmospheric effects may be included in this category.

The sources of angle error are summarized in Table 2.4 and the sources of range error are summarized in Table 2.5. The components of range and angle error combine to give velocity error in the final solution.

Table 2.4. Classification and Description of Angle Errors.

Error	Bias	Noise
Radar-Dependent Tracking Error	-Boresight axis setting and drift -Torque due to wind/gravity -Servo unbalance and drift	-Thermal -Multipath -Servo -Torque -Deflection due to acceleration of antenna
Radar-Dependent Translation Errors	-Pedestal leveling -Azimuth alignment -Orthogonality of axis -Pedestal flex due to gravity or heating	-Bearing wobble -Data gear non-linearity and backlash -Data takeoff non-linearity and granularity -Pedestal deflection due to acceleration
Target-Dependent Tracking Error	-Dynamic lag	-Dynamic lag -Glint -Scintillation -Beacon modulation
Propagation Error	-Average tropospheric refraction -Average ionospheric refraction	-Variation in tropospheric refraction -Variation in ionospheric refraction

Compiled from (1:326)

2.5.2 Propagation Effects. The propagation of the electromagnetic signal is a problem for radar systems just as it is for communication systems. In fact, some state that the radar system is just an extension of a communication system (29:16). Because of the

Table 2.5. Classification and Description of Range Errors.

Error	Bias	Noise
Radar-Dependent Tracking Error	-Zero range setting -Range discriminator shift -Receiver delay	-Thermal -Multipath -Servo -Variation in receiver delay
Radar-Dependent Translation Error	-Range oscillator (velocity of light) -Data take-off zero setting	-Range resolver -Internal jitter -Data gearing -Data take-off -Range oscillator stability
Target-Dependent Tracking Error	-Dynamic lag -Beacon delay	-Dynamic lag -Glint -Scintillation -Beacon jitter
Propagation Error	-Average tropospheric refraction -Average ionospheric refraction	-Variation in tropospheric refraction -Variation in ionospheric refraction

Compiled from (1:373)

importance of the propagation process, the causes of error resulting from the process are further detailed below.

2.5.2.1 Attenuation. The basis of attenuation calculations in the troposphere are based on the original work of Van Vleck in 1947. The result of Van Vleck's analysis were equations which provided absorption per distance as a function of wavelength, pressure, temperature, and type of gas in the atmosphere (1:468).

The greatest attenuation is caused by clouds and rain due to the effects of water vapor and oxygen. For X-band radar (10,000 MHz), the attenuation may be as great as 1 dB per mile. However, for C-band (5000 MHz), the attenuation is only one-eighth as great (1:468). Studies by Millman in 1958 showed that radars operating above 100 MHz will not suffer attenuation greater than approximately 1 dB even under the worst conditions of ionospheric density (daytime values) (1:470-471).

An additional problem with water vapor and oxygen is the ability of the molecule to extract energy from the radar wave. Since atmospheric molecules are able to extract energy from a radar wave, they are also able to emit energy at radar frequencies. This

emission is received as noise in the radar receiver and is characterized by the temperature of the atmospheric molecule (1:473–474).

2.5.2.2 Surface Reflection. The effect of surface reflection has received much attention in the study of radar effects. The results of this work indicate a coefficient of polarization remains near unity for horizontal polarization for elevations near 0 degrees. For vertical polarization, the coefficient may vary significantly until the system reaches an elevation of 10 degrees. In general, there are no rules for determining an estimate of the reflection coefficient. A value of 0.3 is used in cases where no other information is available. The impact of reflection must be evaluated when the angle of incidence is below a minimum given by the following equation (1:475):

$$E_{min} = \frac{\sqrt{h_{ft}}}{4000} \text{radians} \tag{2.6}$$

where h_{ft} is the altitude of the target in feet. For a satellite with an altitude of 200 kilometers, $E_{min} \approx 11^{o}$.

Another aspect of surface reflection is clutter. Clutter is any scattering element which interferes with radar system operation (1:95). The major impact of clutter is generated by the side-lobe pattern of the radar. This error is radar-design-dependent and based on the antenna construction and transmitted power.

2.5.2.3 Tropospheric Refraction. The effect of tropospheric refraction on radio communication paths is often referred to as the "4/3 earth's radius correction." Tropospheric refraction has the effect of increasing the line of sight path length beyond the geometrical limits. Early radar calculations used the same methodology with good results for objects below 50,000 feet. The current process relies on the exponential reference atmosphere described by the National Bureau of Standards (1:477–478).

The refractive index of air (for frequencies below approximately 20,000 mc) may be described by the Smith-Weintraub constants in the following equation:

$$N = (n-1) \times 10^6 = \frac{77.6}{T}\left(P + \frac{4810p}{T}\right) \qquad (2.7)$$

where T is the atmospheric temperature in degrees Kelvin, P is the total pressure of the atmosphere in millibars, p is the partial pressure of the water vapor component, n is the refractive index, and N is a scaled up value referred to as the "refractivity" (1:477-478). Refractivity of the atmosphere has been shown to vary by the time of the day and time of the year. Changes of 20 to 40 N units are common for diurnal cycles. This implies the refractivity of a single site may change by more than 100 N units over the course of a year (1:481–482).

The refraction of the troposphere causes an extra time delay in the transmission of the signal and an increase in the elevation angle measured by the antenna. The ray follows the path of "minimum delay" to get to the target and return. The errors in range and elevation have been found to be proportional to the refractivity along the measurement path of the ray (1:479).

The National Bureau of Standards describes a way to calculate the total bending of a ray based on the equation below:

$$\Delta E_t = E_o - E_c = (bN_s + a) \times 10^{-6} \text{radians} \qquad (2.8)$$

where E_o is the angle that the ray arrives at the earth, E_c is the angle the ray would have arrived at had there been no atmosphere, and ΔE_t represents the total bending of the ray. The values b and a are functions of the elevation angle E_o. For an elevation above 5°, b is approximately the cotangent of E_o and a is approximately 0. Therefore (1:479-481):

$$\Delta E_t \approx N_s \cot E_o \times 10^{-6} \text{ radians} \qquad (2.9)$$

The temporal fluctuations in the range, elevation, and azimuth of the target are caused by variation in the troposphere drifting through the beam of the radar. These errors are in addition to the regular bias errors caused by the stratified nature of the

atmosphere. Based on the work of Muchmore and Wheeler, the standard deviation of the range (σ_r) and angle (σ_a) measurements may be calculated from:

$$\sigma_r = \sqrt{2l_o L \Delta\overline{N^2}} \times 10^{-6}\text{radians} \tag{2.10}$$

$$\sigma_a = \sqrt{\Delta\overline{N^2} L / l_o} \times 2 \times 10^{-6}\text{radians} \tag{2.11}$$

In these equations, l_o is the scale length of the tropospheric anomalies causing the fluctuation, L is the path length containing the anomalies, and $\Delta\overline{N^2}$ is the mean square refractivity variation of the anomalies (1:485-486).

2.5.2.4 Ionospheric Refraction. The effects of the ionosphere must be taken into account for targets over 60 to 70 miles above the surface. The refraction effects are all dependent on the operating frequency of the system and vary directly with the square of the wavelength (1:490).

The refractivity of the ionosphere may be written as a function of the radar frequency:

$$N_i = (n-1) \times 10^6 = \sqrt{\frac{N_e e^2}{\epsilon_o m \omega^2}} \times 10^6 \cong \frac{1}{2} \frac{N_e e^2}{\epsilon_o m \omega^2} \tag{2.12}$$

where N_e is the electron density per m^3, e is the charge of the electron, m is the electron mass, ω is the frequency, and ϵ_o is the permitivity of free space. If you insert the values for the known constants, this equation reduces to:

$$N_i \cong 40 \left(\frac{N_e}{f^2} \right) = \frac{1}{2} \left(\frac{f_c}{f} \right)^2 \tag{2.13}$$

where f is the frequency of the radar and $f_c = 9\sqrt{N_e}$ is the critical frequency of the medium (1:491).

2.5.3 Optical Sensor Error. Like the radar sensor, the function of an optical sensor is to acquire the target, and track it. Again, like the radar, the physical limits of the optical system impact its ability to acquire, track, and point.

For optical systems, tracking is the movement of the instrument line of sight to follow the movement of the target. Pointing is the average direction of the instrument line of sight. Therefore, tracking error is a time-varying component which measures the difference in how the sensor should follow the target and how it actually follows the target. Pointing error is the static component which measures the difference in where the instrument should be pointing versus where it is pointing. To obtain a total error, these components of error are averaged over the dwell time of the track[4] (32:40).

2.5.3.1 Acquisition Error. The acquisition performance of the sensor system is aided by the appropriate design of the CCD and lens subsystems. The target may be detected as long as its image occupies one or more picture elements (pixels) of the CCD. However, this assumes the CCD is flawless. Inoperable pixels may result in the target being missed or detected too late to accurately track (32:44). The missed or late target track will severely corrupt the data for orbit element determination.

The other general sources of acquisition error are air turbulence, visual magnitude of the target, and background radiation. The turbulence of the atmosphere degrades the sensor ability to acquire the target. Turbulence increases the effects of refraction and introduces deviations to the azimuth and elevation measurements of the target. Visible magnitude is the apparent brightness of the target. If the target is not bright enough to register on the CCD, a missed or late target error may occur. Background radiation also impacts target magnitude. If the background is brighter than the target, the target will "wash out" and not be visible (32:45).

A general factor of all electronic equipment is the emission of random thermal noise. Thermal noise is the generation of electrons due to the temperature of the equipment. For a CCD, these extra electrons will be translated into false images for very brief periods of time. It is possible to eliminate this random error by observing the target for long periods of time. However, the time available for tracking may not be sufficient if you are attempting to observe a low-earth satellite passing quickly through the field of view (32:47).

[4]The required observation time necessary for the mission.

2.5.3.2 Tracking Error. Tracking is the stabilization of the line of sight of the system while following the target. Tracking is measured by a separate sensor which detects the difference in the sensor boresight and the sensor line-of-sight. For precise tracking, the allowable error is measured against predetermined object size. If the target passes through the allowable range of the predetermined size, the tracking mechanism moves the sensor to place the target into an optimal position. The measure of this motion is called response time. Response time is based on the acceleration rate and constant velocity ability of the sensor (32:50).

Gimbals provide the support, stability, and movement for the optical sensor. Gimbal performance may be evaluated from four areas: angular range, balance, stiffness, and bearing size. Angular range refers to the azimuth (compass) motion and elevation (measured from horizon to zenith) motion of the gimbal. The desired azimuth range is 360 degrees, but 270 degrees is acceptable for most tracking applications. Elevation should cover the full range of 0 to 90 degrees. The gimbal should have its mass equally distributed to provide stability. The stiffness and bearing size are design variables which depend on the overall system requirements (4:116).

In general, the tracking errors are independent of the gimbal size. However, smaller systems are quicker and easier to control and are more capable of rejecting undesirable disturbances. On the other hand, large systems are harder to disturb. Other errors may be limited by spreading the image over two or more pixels. This would reduce the movement of the system. However, the size of the image is limited by the physical characteristics of the optics and the range to the target. For most satellite tracking applications, a satellite's dimensions are very small in comparison to the altitude. Therefore the size of the image may not cover multiple pixels.

Various disturbance errors such as wind, seismic forces, gyroscope noise, and electronic noise have all been measured or computed. They may be factored into the calculations for position and subtracted to obtain the final answer. The typical tracking error for advanced telescopes is a standard deviation of less than 500 nanoradians (32:53). Tracking rate is the velocity the sensor must swivel to follow the target. The faster the target moves, the sooner the target leaves the sensor field of view. The major causes of error

in this component are the forces acting on the gear and drive mechanism in the gimbal. These forces are defined as friction, coulomb, and viscous.

The friction force (sometimes called stiction) is the torque required to break loose the gears and start motion. The coulomb force is a constant force developed from the motion of the gears and is independent of motion. The viscous force is directly dependent on the rate of the gear motion. The effects of the forces may be overcome through the use of hydrostatic oil bearings and compressed air bearings where high angular rates are required (i.e., for a low-earth satellite). The drawback to these types of bearings are their high cost to manufacture and maintain (32:57). As an additional improvement in gimbal operation, new instrumentation techniques are available to provide inertial control to the gimbal (5:154).

2.5.3.3 Pointing Error. Pointing is the requirement to maintain a constant direction value for azimuth and elevation for the sensor. Pointing error is a static measurement taken when the sensor is not moving. For most satellite tracking applications, this is not applicable (except for geostationary orbits observed from the equator). The current system requirements are for less than a 5 microradian difference above 20 degrees in elevation measurement. The key to pointing control is in the thermal control of the gimbal device.

All metals expand when heated. When the gimbal structure heats up and expands, small errors in pointing are detected. The key is to control the expansion through insulation or active cooling (32:60).

For overall system performance, various manufacturing errors may be detected and subtracted out since they will remain constant with time. Common gimbal errors include axis misalignment and incorrect mass allocation of the gimbal structure (heavier on one side than the other).

2.5.4 Sensor Bias and Sigma. All of these effects combine to create unique errors for each sensor which vary for each observation. These errors must be removed from the observations if an accurate correction to the element set is to be obtained. Most

major known sources of error are typically removed by the individual sensors prior to forwarding the observations to the SSC. However, the SSC must still track and account for any remaining errors.

The SSC uses a simple method to quantify errors. Sensor observations are gathered on a satellite for which highly accurate orbital data is known. The observations are compared to predicted observations using the accurate orbital data and the residuals calculated. The residuals are statistically analyzed and the mean error, or bias, and the standard deviation of the error, or sigma, are computed (11:4-1) (36:76,239) (24:205).

There are two methods used for generating the highly accurate orbital data. The first is using the output from a special perturbations correction to generate accurate ephemeris from which the accurate predicted observations can be obtained (24:205). The second is to use ephemerides from an outside agency such as the Defense Mapping Agency (DMA), GPS, or the Naval Astronautics Group (NAG) (24:205). A common calibration satellite used is a Navy Transit Navigation System satellite.[5] This satellite has an onboard transponder which is used to obtain highly-accurate positional data. Ephemerides obtained from DMA for this satellite have an accuracy on the order of five meters (11:4-2).

The SSC tasks all sensors to routinely gather observations on calibration satellites. An SSC program called SPCALX is executed daily to calculate the sensor biases and sigmas (24:205) (36:239 241). These values are stored in the LOWB file for use in processing the observations and correcting the elements (24:206.1). They are also routinely monitored by SSC personnel for degradations in the sensors.

Tables 2.6 and 2.7 provide a representative sample of observation sigmas and biases for each of the sensors of the SSN. The majority of these values are compiled from the 22 September 92 daily weights and biases message released by the SSC. When data was not included in the message for a particular sensor, data from the November 1988 AFSPACECOM/DOA Metric Accuracy Study was used (11:4-2 to 4-6).

[5]For example, NORAD Catalog Number 6909.

Table 2.6. Representative Sensor Sigmas.

Sensor Number(s)	Sensor Name	Azimuth or Rt Ascension* Sigma degrees	Elevation or Declination* Sigma degrees	Range Sigma km	Range Rate Sigma km/sec
211, 212, 213	GEODSS SOCORRO	0.005	0.005	Note 1	Note 2
221, 222, 223	GEODSS TAEGU	0.005	0.005	Note 1	Note 2
231, 232, 233	GEODSS MAUI	0.005	0.005	Note 1	Note 2
241, 242, 243	GEODSS DIEGO GARCIA	0.005	0.005	Note 1	Note 2
333	ALCOR[†]	0.011	0.011	0.090	0.0007
334	ALTAIR	0.011	0.011	0.090	0.0007
337, 401	PIRINCLIK[‡]	0.022	0.021	0.028	0.0035
403, 404	PIRINCLIK	0.055	0.041	0.500	0.0002
344, ???, ???	FYLIKGDALES[§]	0.028	0.027	0.056	0.0019
349, 359	CLEAR[‡]	0.045	0.021	1.511	0.0019
354, 355	ASCENSION	0.007	0.006	0.074	0.0065
363	ANTIGUA	0.009	0.009	0.070	0.00[illegible]
369	MILLSTONE HILL	0.005	0.005	0.005	0.0001
370	MILLSTONE UHF[¶]	0.005	0.005	0.005	0.0001
382, 383	GOODFELLOW[‡]	0.029	0.027	0.059	0.0020
384, 385	ROBINS[‡]	0.045	0.034	0.064	0.0025
386, 387	CAPE COD[‡]	0.022	0.025	0.069	0.0021
388, 389	BEALE[‡]	0.020	0.020	0.042	0.0020
393	COBRA DANE	0.014	0.009	0.032	0.0012
394, 395	THULE[‡]	0.021	0.028	0.043	0.0011
396	PARCS	0.006	0.014	0.035	0.0012
398	EGLIN	0.036	0.045	0.700	Note 2
399	EGLIN	0.015	0.018	0.036	Note 2
741 747	NAVSPASUR (745)	0.012	0.010	0.358	Note 2
932	KAENA POINT	0.004	0.004	0.039	0.0008
951	MOTIF[‖]	0.005	0.005	Note 1	Note 2
952	AMOS[‖]	0.005	0.005	Note 1	Note 2

Note 1: Sensor does not report range data.
Note 2: Sensor does not report range rate data
* Right ascension and declination reported for optical sensors only.
† No data available. For this research, sigmas assumed same as sensor 334.
‡ Sigmas are average of multiple co-located or near co-located sensors.
§ Estimated sigmas for new three-face phased-array radar based on average sigmas of all PAVE PAWS sensors (382 thru 389, 394, and 395).
¶ No data available. For this research, sigmas assumed same as Sensor 369
‖ No data available. For this research, sigmas assumed same as GEODSS sensors

Compiled from (34) and (11 4-6)

Table 2.7. Representative Sensor Bias.

Sensor Number(s)	Sensor Name	Azimuth or Rt Ascension* Bias	Elevation or Declination* Bias	Range Bias	Range Rate Bias
		degrees	degrees	km	km/sec
211, 212, 213	GEODSS SOCORRO[†]			Note 1	Note 2
221, 222, 223	GEODSS TAEGU[†]			Note 1	Note 2
231, 232, 233	GEODSS MAUI[†]			Note 1	Note 2
241, 242, 243	GEODSS DIEGO GARCIA[†]			Note 1	Note 2
333	ALCOR[†]				
334	ALTAIR	0.098	0.012	0.075	-0.0001
337, 401	PIRINCLIK[‡]	-0.003	-0.002	-0.035	-0.0008
403, 404	PIRINCLIK				
344, ???, ???	FYLINGDALES[‡]				
349, 359	CLEAR[‡]	-0.017	0.014	1.133	0.0048
354, 355	ASCENSION	0.003	0.006	0.017	-0.0005
363	ANTIGUA	-0.006	-0.003	0.000	0.0005
369	MILLSTONE HILL	0.002	0.003	0.002	0.0000
370	MILLSTONE UHF[†]				
382, 383	GOODFELLOW[‡]	0.013	-0.022	0.017	0.0000
384, 385	ROBINS[‡]	0.006	0.026	0.020	0.0004
386, 387	CAPE COD[‡]	0.011	-0.004	-0.002	0.0002
388, 389	BEALE[‡]	-0.019	0.000	0.053	-0.0031
393	COBRA DANE	0.014	-0.006	-0.038	0.0001
394, 395	THULE[‡]	-0.011	-0.040	0.009	-0.0004
396	PARCS	0.004	-0.024	-0.030	0.0003
398	EGLIN[†]				Note 2
399	EGLIN	-0.009	0.008	0.023	Note 2
741 747	NAVSPASUR (745)	-0.002	-0.011	-0.358	Note 2
932	KAENA POINT	0.001	-0.004	0.004	0.0000
951	MOTIF[†]			Note 1	Note 2
952	AMOS[†]			Note 1	Note 2

Note 1 Sensor does not report range data

Note 2 Sensor does not report range rate data.

* Right ascension and declination reported for optical sensors only

† No data available

‡ Biases are average of multiple co-located or near co-located sensors

Compiled from (34) and (11 4 6)

2.6 *Orbit Classifications*

There are two reasons for creating orbital classes. The first is a shorthand to make communication easier; informal classes serve this purpose. The second reason is because they share some characteristic that makes them worthy of a class. Gabbard classes divide the satellite population into similar tasking characteristics.

2.6.1 Informal Classes. Since the Space Surveillance Center's orbit propagation models handle orbits with periods less than 225 minutes differently than those greater than or equal to 225 minutes, a satellite is typically classified as near-earth or as deep-space based on its period (16:1). In addition to the near-earth or deep-space classification, other types of orbits are often considered as classes. For instance, geosynchronous satellites have periods that match the earth's rotation rate. Geostationary satellites are geosynchronous satellites that remain above the same sub-point on the earth's surface.

Besides period classifications there are also inclination classes. A prograde orbit has an inclination less than 90 degrees while a retrograde orbit has an inclination greater than 90 degrees. Zero inclination orbits are termed "equatorial." Less obvious, but no less important, are the two critical inclinations. Satellites at a critical inclination have a stationary argument of perigee. Other satellites are termed sun-synchronous because the ascending node regresses at the same rate as the earth's angular rate about the sun.

2.6.2 Gabbard Classes. For tasking, the SSN classifies satellites using what are called the Gabbard classes. There are 32 Gabbard classes. The 32 Gabbard classes divide the satellite population according to perigee and apogee altitude. The classes are one of the factors used by the SSN for tasking the network's sensors. Table 2.8 lists the actual classes and the perigee and apogee limits for each of the classes. Figures 2.2 through 2.4 provide a method to visualize each of the 32 classes.

The reasons for the particular classes are (25:144):

1. The altitude below 575 kilometers is a region where satellites will experience relatively severe atmospheric drag. Eccentric orbits of small debris will commence to circularize when perigee drifts below this limit.

Table 2.8. Gabbard Classes.

Gabbard Class	Apogee Altitude		Perigee Altitude	
	Min	Max	Min	Max
	km	km	km	km
1	0	575	0	575
2	575	1000	0	575
3	575	1000	575	1000
4	1000	2000	0	575
5	1000	2000	575	1000
6	1000	2000	1000	2000
7	2000	3000	0	575
8	2000	3000	575	1000
9	2000	3000	1000	2000
10	2000	3000	2000	3000
11	3000	5555	0	575
12	3000	5555	575	2000
13	3000	5555	2000	3000
14	3000	3700	3000	3700
15	3700	5555	3000	3700
16	3700	5555	3700	5555
17	5555	35000	0	575
18	5555	35000	575	2000
19	5555	35000	2000	3000
20	5555	35000	3000	3700
21	5555	35000	3700	5555
22	5555	11110	5555	11110
23	11110	35000	5555	11110
24	11110	35000	11110	35000
25	35000	∞	0	575
26	35000	∞	575	2000
27	35000	∞	2000	3000
28	35000	∞	3000	3700
29	35000	∞	3700	5555
30	35000	∞	5555	11110
31	35000	∞	11110	35000
32	35000	∞	35000	∞

Compiled from (25:143-144)

2. The altitude area between 575 and 1,000 kilometers contains atmosphere of tenuous density but sufficient to noticeably perturb general perturbations orbits.

3. The altitude area above 1,000 kilometers does not contain enough atmosphere to be a factor in general perturbations orbit predictions.

4. The altitude limits of 2,000, 3,000, 5,555, and 11,110 kilometers provide convenient references to qualitative differences in the effect of eccentricity or to radar sensor single and double-bang range capabilities.

5. The limit at 35,000 kilometers is slightly below true circular synchronous altitude and was selected instead of true synchronous to ensure that Class 32 would contain all nearly-circular synchronous satellites.

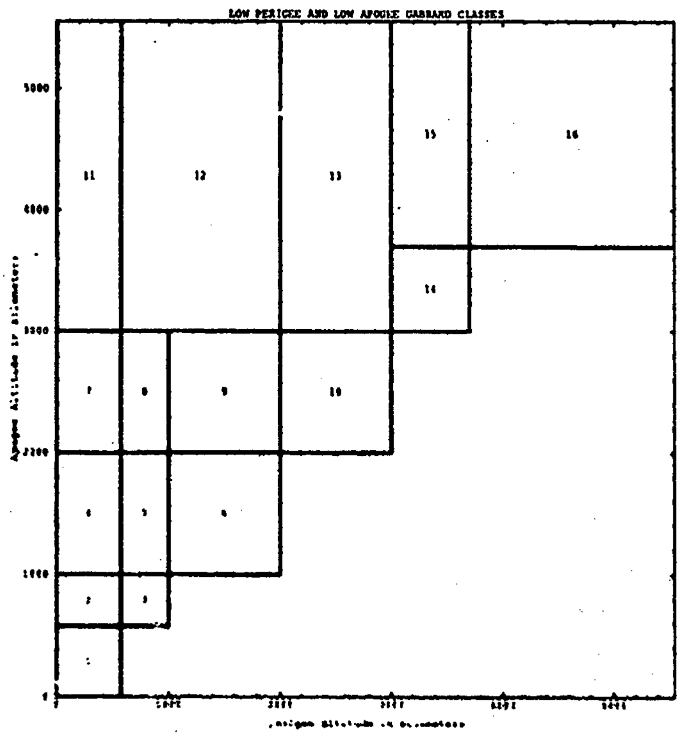


Figure 2.2. Near-Earth-Orbit Gabbard Classes.

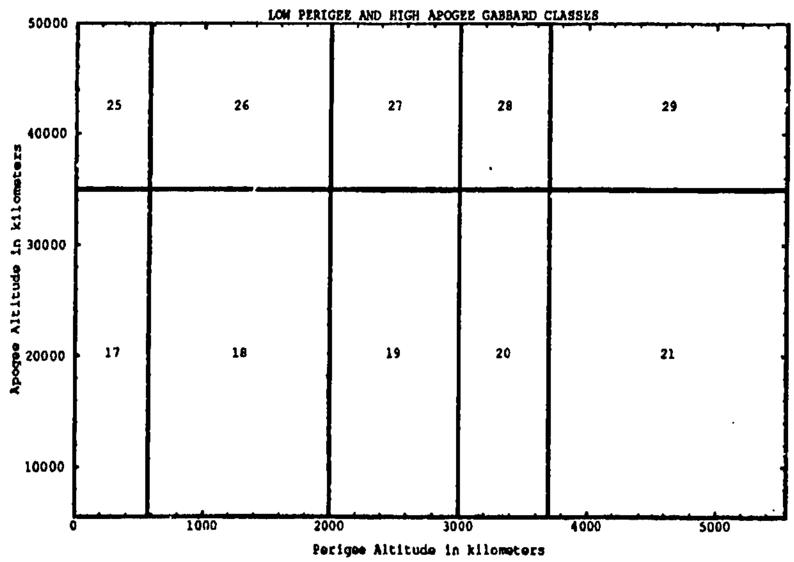


Figure 2.3. Deep-Space Gabbard Classes (High Eccentricity).

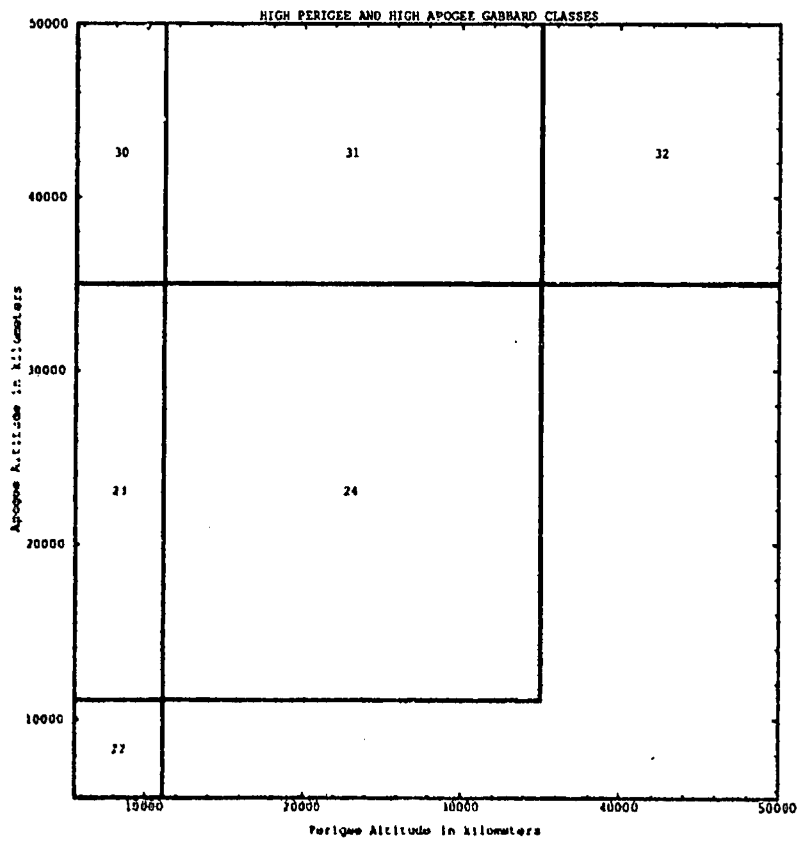


Figure 2.4. Deep-Space Gabbard Classes.

2.7 *Sensor Tasking Methodology*

The sensors of the SSN are tasked by the SSC to gather observations on all earth-orbiting satellites. In an attempt to make the most efficient use of SSN and SSC resources, a method called "Selective Tasking" is used (35:3-10). The three goals of Selective Tasking, as defined by USSPACECOM Regulation 55-12, are:

- Obtain the correct number and dispersion of observations for each satellite.
- Ensure the most efficient use of the Space Surveillance Network.
- Ensure that the observations on high-interest satellites are obtained and forwarded on a priority basis.

The three basic underlying concepts of Selective Tasking, as defined by USSPACECOM Regulation 55-12, are:

- Observation Regulation. Sets upper and lower limits on the number of observations collected on a given satellite. Controlling the amount of data gathered ensures the most efficient use of the SSN.
- Observation Requirements. Observation requirements are dependent on the orbital parameters and mission of the satellite. A routine satellite in a stable orbit will require less observations than a high-interest, maneuverable satellite.
- Observation Dispersion. Observations spread out around the entire orbit are more useful than the same number of observations clustered in one area.

Observational data requirements in terms of the number, quality, dispersion, and priority of observations will vary based on a satellite's mission, orbital parameters, and by the sensors capable of tracking the satellite. Orbital analysts in the SSC use three programs, called RTASK, DSTASK, and BNPLAN, to determine the routine tasking requirements for all satellites. As stated in NORAD Technical Publication 008,[6] RTASK uses the results from a "System Capability Study" performed between September 1973 and April 1974 to determine the number observations required for all near-earth satellites (24:212,223). The

[6]Dated 6 April 1992, last updated 8 October 1991.

results of this study, as shown in Table 2.9, were implemented in 1974 and have been in use since that time (24:212). We found no references indicating that observation requirements from a more recent study were in use.

As seen in Table 2.9, no observation requirements were set for Gabbard Classes 19 through 32 (deep-space classes). This was because of the lack of sensor coverage for deep-space objects at that time and, therefore, a lack of a sufficient number of observations to support this empirical study. The operational requirement for these classes is 30 observations per LUPI (length of update interval), where LUPI equal to 30 days for non-synchronous satellites and 60 days for synchronous satellites (24:213). The programs DSTASK and BNPLAN are used to recommend tasking for these deep-space objects. DSTASK is used to determine the actual tasking for all deep-space sensors (36:75). BNPLAN then uses the tasking set by DSTASK to generate an observing schedule for optical sensors (36:75).

These three programs are periodically executed (typically daily) to determine required tasking. In addition, tasking monitoring programs also keep track of the number of observations received compared with the tasking level. This program flags any satellites which are not getting the required quantity of observations (24:225–226). SSC orbital analysts can make any necessary changes to this tasking based on current mission requirements. This sensor-specific tasking is then forwarded to all sensors of the SSN using a system of numeric tasking categories and alphabetic tasking suffixes to describe the observation data requirements for each satellite. The tasking categories and suffixes are also used to regulate the flow of observations from the sensors of the SSN to the SSC.

2.7.1 Tasking Categories and Suffixes. Tasking categories define the priority of getting observational data on the satellite and the precedence for transmitting that data to the SSC. They are also used to resolve conflicts when multiple satellites requiring observational data are in the sensors field of view at the same time and the sensor is unable to gather the required amount of data on all satellites. Tasking suffixes define the actual amount of observational data that must be gathered for the satellite and the frequency of data collection. The one exception to this definition of category and suffix combination is

Table 2.9. 1974 System Capability Study Results.

Gabbard Class	Satellite Type	LUPI* (Days)	Required Obs Per LUPI	Average Obs/day
1	Payload	7	30	4.3
1†	Rocket	6	20	3.3
1†	Debris	5	+30	+6.0
2†‡	Payload	7	20–30	2.9–4.3
2	Rocket	7	20	2.9
2†	Debris	6	+30	+5.0
3	Payload	11	12	1.1
3	Rocket	10	13	1.3
3	Debris	9	13	1.4
4†	Payload	6	30	5.0
4†‡	Rocket	7	30	4.3
4†	Debris	7	20	2.9
5	Payload	11	14	1.3
5	Rocket	11	16	1.5
5	Debris	11	15	1.4
6	Payload	13	12	0.9
6	Rocket	13	16	1.2
6	Debris	14	17	1.2
7†	Payload	8	20	2.5
7	Rocket	7	14	2.0
7	Debris	8	15	1.9
8‡	Payload	10	16	1.6
8	Rocket	9	15–20	1.7–2.2
8	Debris	10	15	1.5
9	Debris	14	15	1.1
11†	Payload	6	30	5.0
11†	Debris	11	+30	2.7
12‡	Payload	10	15–20	1.5–2.0
12‡	Rocket	10	20	2.0
12†‡	Debris	17	+30	1.8
13†	Debris	16	+30	1.9
14†	Payload	13	15	1.2
14†	Debris	14	15	1.1
15†	Payload	14	20	1.4
15†	Debris	15	30	2.0
18†	Payload	17	30	1.8
18†	Rocket	13	30	2.3
19–32†	N/A	N/A	Note 1	Note 1

Note 1: No data obtained.
* LUI = Length of Update Interval.
† Test sample small, data erratic, or both.
‡ Did not meet 12 km vector magnitude accuracy goal regardless of observation rate tested.

Compiled from (24:212–213)

the NAVSPASUR sensor. It uses only tasking categories to define both the priority and amount of observational data required.

2.7.2 Near-Earth Sensor Tasking. The categories used with near-earth mechanical and phased-array radars are (36:73):

- Category 1: Events of the highest priority. Used for satellites requiring instantaneous observational data (i.e., new foreign launches, satellites in final stages of decay, and near-earth satellites performing maneuvers). Data is sent by IMMEDIATE precedence unless FLASH precedence is requested by the SSC.
- Category 2: Special events of high priority. Satellites of high priority but not requiring instantaneous data (i.e., satellites capable of performing maneuvers, de-orbiting satellites, and domestic launches). Data is sent by PRIORITY precedence unless IMMEDIATE precedence is requested by the SSC.
- Category 3: All routine satellites. Data sent by ROUTINE precedence unless otherwise requested.

Tables 2.10 and 2.11 describe the tasking suffixes used with near-earth mechanical and phased-array radars.

As mentioned above, NAVSPASUR uses categories alone to define both the priority and amount of observational data required. Those categories are (36:73-74):

- Category 1: Same as Category 1 for mechanical and phased-array radars above.
- Category 2: Same as Category 2 for mechanical and phased-array radars above.
- Category 3: Satellites with orbital parameters making them difficult to track and, therefore, requiring continuous observations. Kickapoo-referenced[7] observations required on all passes. Data is sent by ROUTINE precedence.
- Category 4: Satellites in stable orbits. One Kickapoo referenced observation from the most westerly pass during each successive 24-hour period. Data is sent by ROUTINE precedence.

[7] Kickapoo is the name of the town where the main NAVSPASUR receiver is located.

Table 2.10. Near-Earth Mechanical-Tracking Radar Tasking Suffixes.

Tasking Suffix	Description
A	Maximum data on all available passes.
B	1 1/2 minutes of data on all available passes.
C	1 1/2 minutes of data on four passes/day.
D	1 minute of data above 15° elevation, centered about culmination, for 2 passes/day separated by 12 ± 4 hours. Used for calibration satellites.
E	1 observation/day. Select portion of pass providing the best quality data.
F	1 1/2 minutes of data on 2 passes/day.
G	1 observation/pass for 2 passes/day. Select portion of pass providing the best quality data. Use 1 ascending and 1 descending pass when possible.
H	3 observations/pass for all available passes.
J	10 observations/pass for all available passes.
S	Special tasking as set by SSC.
T	Final TIP tasking; maximum data on all available passes.

Compiled from (36:78)

Table 2.11. Near-Earth Phased-Array Radar Tasking Suffixes.

Tasking Suffix	Description
A	Maximum data on all available passes.
B	15 observations/pass on all available passes.
C	5 observations/pass on all available passes.
D	3 observations/pass on all available passes.
E	1 observations/pass on all available passes.
F	1 observation/day.
G	2 observations/day.
H	3 observations/day.
J	4 observations/day.
K	1 observation/day on an ascending pass.
L	1 observation/day on a descending pass.
M	2 observations/day. 1 ascending pass and 1 descending pass.
P	9 observations/pass above 7° elevation, for 2 passes/day. separated by 12 ± 4 hours. Used for calibration satellites.
S	Special tasking requirements as set by SSC.
T	Final TIP tasking; maximum data on all available passes.

Compiled from (36:79)

- Category 5: Satellites of special importance. Observation requirements as set by the SSC. Data is sent by ROUTINE precedence.
- Category 6: Satellites requiring low data rates. One Kickapoo-referenced observation every 36 hours alternating between ascending and descending passes. Data is sent by ROUTINE precedence.

Each sensor tasked to gather observations on near-earth satellites receives a weekly tasking message from the SSC. This message lists that sensor's tasking requirements for all near-earth satellites (36:68). Updates to this message are made by the SSC as requirements and mission dictate.

For the majority of the satellite population (routine near-earth satellites), only four sensors are tasked to gather observations (3). This tasking is presented in Table 2.12. If these sensors are not able to gather the minimum required observations, or if circumstances require additional observations, other sensors will be tasked based on the following Sensor Tasking Priority List (12:Appendix 2) (3):

745, 399, 393, 382, 363, 354, 331, 932, 337, 349, 342, 394, 388, 386, 384

Table 2.12. Typical Tasking for Routine Near-Earth Satellites.

Sensor Number	Sensor Name	Tasking
745	NAVSPASUR	Category 3 for all satellites with inclinations > 30°.
399	EGLIN	Category 3H for all satellites.
396	PARCS	Category 3H for all satellites with inclinations ≥ 30°.
382	ELDORADO	Category 3H for all satellites with inclinations < 50°. Category 3F for all satellites with inclinations ≥ 50°.

Compiled from (12:2)

2.7.3 Deep-Space Sensor Tasking. The categories used with all sensors that track deep-space satellites are (36:74):

- Category 1: Same as Category 1 for mechanical and phased-array radars above.
- Category 2: Same as Category 2 for mechanical and phased-array radars above.

- Category 3: Routine satellites and satellites that have not been tracked in the last 4 to 30 days. Data sent by PRIORITY precedence.
- Category 4: Satellites that have not been tracked in the last 2 to 4 days. Data sent by ROUTINE precedence.
- Category 5: Satellites that have not been tracked in last 1 to 2 days. Data sent by ROUTINE precedence.

Tables 2.13 and 2.14 describe the tasking suffixes used with deep-space radar and optical sensors.

Table 2.13. Deep-Space Optical Sensor Tasking Suffixes.

Tasking Suffix	Description
A	Maximum data on all available passes.
B	15 observations/pass on all available passes.
C	4 observations/pass on 1 pass/day.
D	4 observations/pass on 2 passes/day separated by at least 15° of true argument of latitude.
E	4 observations/pass on 2 passes/day separated by at least 30° of true argument of latitude.
F	3 observations/pass on 1 pass/day. Used for calibration satellites
M	4 observations/pass on a minimum of 2 passes/day to verify position. If object not detected or orbit deviation suspected, change to S.
N	4 observations/pass on 1 pass/day every even-numbered days.
O	4 observations/pass on 1 pass/day every odd-numbered days.
Q	4 observations/pass on 1 pass/day every 5 days.
R	Special tasking as set by the SSC.
S	An in-track or cross-track search once/day. When object found, change to suffix E.
T	Final TIP tasking. 3 observations/pass on 3 passes/day. If object not detected, change to suffix S.

Compiled from (36:84–86)

Each sensor tasked to gather observations on deep-space satellites receives a daily tasking message from the SSC. This message lists the sensor's tasking requirements for all deep-space satellites (36:68). Updates to this message are made by the SSC as requirements and mission dictate.

Table 2.14. Deep-Space Radar Sensor Tasking Suffixes.

Tasking Suffix	Description
A	Maximum data on all available passes.
B	15 observations/pass on all available passes.
C	4 observations/pass on 1 pass/day.
D	4 observations/pass on 2 passes/day separated by at least 1 hour.
E	4 observations/pass on 2 passes/day separated by at least 2 hours.
F	3 observations/pass on 1 pass/day. Used for calibration satellites
M	4 observations/pass on a 2 passes/day to verify position. If object not detected or orbit deviation suspected, maximum data required.
N	4 observations/pass on 1 pass/day every even-numbered days.
O	4 observations/pass on 1 pass/day every odd-numbered days.
Q	4 observations/pass on 1 pass/day every 5 days.
R	Special tasking as set by the SSC.
S	An in-track or cross-track search once/day. When object found, maximum data required.
T	10 observations/pass on all available passes.

Compiled from (36:84-86)

2.8 SSN Sensor Observation Processing Methodology

The SSC receives observations from a group of diverse types of sensors. Some of the sensors were originally designed for other missions and were adapted to the SSN because of their capabilities. Because of these differences, not all sensors report their observations in the same format. Table 2.15 describes all observation types that the SSC is programmed to handle.

Table 2.15. Observation Data Types.

Observation Type	Observational Data Gathered by the Sensor
0	Time, range rate.
1	Time, azimuth, elevation.
2	Time, azimuth, elevation, range.
3	Time, azimuth, elevation, range, range rate.
4	Time, azimuth, elevation, range, range rate, azimuth rate, elevation rate, range acceleration
5	Time, right ascension, declination.
6	Time, range.

Compiled from (24:211)

Of the sensors currently comprising the SSN, all mechanical-tracking and phased-array radar sensors, with the exception of EGLIN, report Observation Type 3. Eglin and NAVSPASUR report Observation Type 2. All optical sites report Observation Type 5.

The processing methodology of observations at each sensor site is site dependent. The sensors respond to taskings generated by RTASK or requested by the analyst at the SSC. Based on the transmitted tasking category and suffix, the internal software at the site selects the best pass and observations to support the tasking (2).

The only software commonality between operations is thought to be among similar sensors. Sensors which achieved initial operational capability in the same time span may process observations with similar software. However, this is not certain since on-site programmers have the ability to reprogram the systems. A recent example of this ability was reported to have improved the processing ability of the sensor at Eglin Air Force Base (22:5).

On an annual basis, sensors of the SSN report more than 32,000 uncorrelated targets (UCTs) to the SSC; however, only 5 percent are actually UCTs (22:5). The reprogramming at Eglin was designed to reduce the number of UCTs by collecting multiple observations before labeling the target as uncorrelated. The effort at Eglin resulted in a reduction of uncorrelated tracks from 175 to 70 and uncorrelated targets from 3500 to 800 (22:5).

2.9 *Observation Processing and Orbital Element Correction*

2.9.1 SSC Observation Processing. All incoming observation data and electronic messages are sent through the system input processor (SIP). The SIP serves as a filter to sort out sensor observations from other electronic messages. All observations are then routed to the POBS program in the SSC (35:5-6). The flow of observational data is shown in Figure 2.5.

The POBS program converts the observation from sensor format to internal SSC format for processing. Additionally, the observations are sorted and error checked. Observations which have no errors other than satellite number are routed to the various other

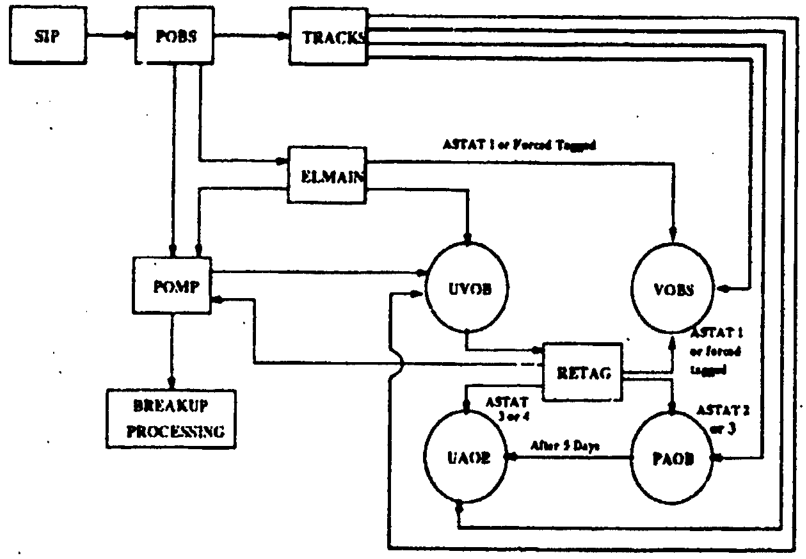


Figure 2.5. Processing Flow of Observations.

programs. From the POBS program, observations are routed to three possible programs based on their "association status": ELMAIN, TRACKS, or POMP (35:5-6).

The association status (ASTAT) of an observation is determined by comparing the observed location of the satellite to its predicted location. To be fully associated (ASTAT 1), the observed location must fall within limits established around the predicted location (24:158). These limits form a volume in space and are referred to as height (Δh radial location), time (Δt in-track location), and plane (β). As limits are exceeded, the association status changes. The limits for these measures are: Δh - 10 kilometers, Δt - 0.05 minutes, and β - 0.05 degrees (24:158). Table 2.16 outlines the ASTAT and limits used by the SSC.

Table 2.16. Association Status.

ASTAT	Limits
1	Meets β, Δh, and Δt
2	Meets β, Within $4 \times \Delta h$ and $4 \times \Delta t$
3	Meets only β
4	β limit exceeded

Compiled from (24:158)

ELMAIN verifies correct satellite number tag by computing residuals and comparing the residuals to acceptable threshold values of plane, time, and height. The purpose of this analysis is to determine the association status of the satellite. The purpose of the association process is to determine the proper "tag" for the observation (24:157). If the satellite has Association Status 1 (ASTAT 1), the observation is routed to the file VOBS (24:159). Observations with ASTAT 2 or 3 are routed to the UVOB file (35:5-6) (24:157). Satellites with ASTAT 4 are routed to the un-associated observations file (UAOB) (24:159).

The TRACKS program combines observations from multiple sensors to create tracks on satellites and build initial element sets. The initial element set is then refined by a differential correction process. Once refined, the element set is compared to the satellite file (SATF) and a degree of correlation is established. If the TRACKS program cannot build an element set, the observations are routed to the UVOB file. Any observations meeting ASTAT 2, 3, or 4 criteria and deep-space object criteria are routed to the VOBS file and the "deep-space hold bucket" (35:5-7).

The POMP program identifies observations which might be associated with a satellite breakup or new launch. POMP uses a plane check to verify the observations. Verified observations are routed to the launch and breakup processor. Observations which are not verified are routed to the UVOB file (35:5-7).

The RETAG program is the next step in processing observations from the UVOB file. This program is automatically initiated every 30 minutes or when there are 140 observations in the UVOB file. RETAG correlates each observation in the UVOB file with each element in the SATF. If an observation correlates to ASTAT 1 or if the satellite is forced tagged to 1, the observation is then routed to the VOBS file. If the correlation is ASTAT 2 or 3 and a trouble flag (see Table 2.17) has been set, the RETAG program routes the observation to the partially associated observation (PAOB) file. If the ASTAT is 3 or 4 and a trouble flag has not been set, the observation is routed to the UAOB file. If an observation is ASTAT 3 and one of the three best candidate satellites has a trouble flag set, RTAG will route the observation to the PAOB file. All observations in the PAOB file that are more than 5 days old are routed to the UVOB file (35:5-8).

Table 2.17. Trouble Flags.

Flag Name	Description
SALAUN	Launch Flag
SAMNVR	Maneuver Flag
SABKUP	Satellite Breakup
SALOST	Lost Satellite Flag
SANDK	Decay Flag

Compiled from (35:5-8)

2.9.2 Differential Correction: Batch and Sequential. Differential correction is a method by which a new estimated state (i.e., the orbital elements) is determined by calculating the state (orbital elements) that "best fits" the orbit based on a set of sensor observations. The SSC uses two methods to differentially correct element sets from received observations: batch and sequential. Table 2.18 outlines the programs used for the types of correction based on the perturbation theory used for the correction.

Table 2.18. Differential Correction Programs.

Sequential		Batch	
Program	Theory	Program	Theory
ELMAIN	General Perturbation	BATCH ELREC FLINT	General Perturbation
STAGEX	Special Perturbation	MANDC	Special or General Perturbation

Compiled from (35:5-9 5-10)

Sequential processing uses the old matrix of elements (GMAT for general perturbation and HMAT for special perturbation) and the new observations or observation track. This procedure results in a time weighting of the observations with more weight being placed on newer observations. Batch processing uses more observations over a longer span than sequential processing. For batch processing, all observations directly contribute to the correction process. There is no time weighting of observations. However, there is a weighting of the sensor observations based on the accuracy of the sensor (35:5-9).

The advantages of sequential correction is the speed of the computer run. However, sequential corrections can only be run on stable orbits since a single bad observation will seriously impact the correction process. A bad observation and, therefore, a bad residual

will have less of an effect on a batch correction because of the larger number of observations used (35:5-10).

2.9.2.1 Automatic Element Set Maintenance. Observations, once processed by ELMAIN or POBS, are placed into holding files. Once the required time has passed or the necessary observations have been collected, the differential correction process begins. However, because of the large number of satellites, it is not feasible to update all of them manually. To provide for automatic updates, there are several programs used to process the observational data (35:5-10). The key element to the process is a table called the satellite control list (SCL).

The SCL is a table in the SCCEX program. The SCCEX program controls the execution and data exchange between all application programs used by the SSC (25:119) (24:204). Within the SCCEX program, the control list is used to control the time sequence of events required to maintain the element sets (35:5-10). To do this, the SCL is divided into two sections. Section 1 is for updating element sets requiring special perturbation and Section 2 is for updating elements requiring general perturbation (35:5-10). The flow of the automated correction process is shown in Figure 2.6.

ELMAIN is the core of the automated differential correction process. This program takes observations and performs a general sequential correction. If the correction is successful, the satellite's GMAT and SATF files are updated. If the correction fails, the satellite is identified for batch correction. ELMAIN will not attempt to correct element sets of satellites which have been tagged for batch correction. All satellites with a HOTF number are tagged for STAGEX processing and are not routed through ELMAIN (35:5-10).

As part of the automated process, the program ELANLX is manually called by the analyst on a regular basis. This program measures the accuracy of the general perturbations element sets. ELANLX compares the predicted position of the satellite with the most current verified observation. If the vector magnitude (VMAG) of the difference (the residual) is greater than 14 kilometers the satellite is selected and marked for BATCH correction in the SCL (35:5-11).

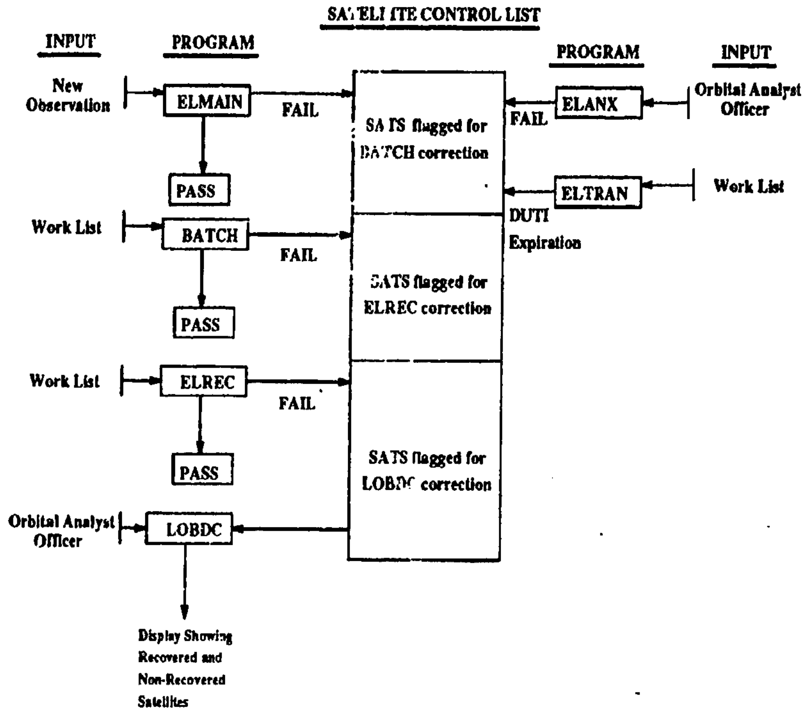


Figure 2.6. Differential Correction Process Flow

An additional automated program is ELTRAN. ELTRAN is time activated based on the expiration of the DUTI number. DUTI is defined as the time interval between runs of BATCH or another correction process (24:203). The typical DUTI for routine satellites is three (indicating three eight-hour periods) (13:Atch 1) (3). If the DUTI has expired, ELTRAN will tag the satellite for BATCH correction (24:204.3).

BATCH is an automatically called program which attempts a general perturbation correction by using only verified observations. If the correction process is successful, the SATF and GMAT files will be updated. If the correction is not successful, the satellite will be tagged for correction by the program ELREC (35:5-12).

ELREC is automatically called based on elapsed time. This program reads satellite numbers from the control list and then attempts a general perturbation correction. However, unlike the BATCH program, ELREC has the ability to access the PAOB and UAOB file and edit observations prior to the correction process. If the correction process is successful, the SATF and GMAT files will be updated. If the process fails, the satellites

are listed on the control list for the lost object differential corrector (LOBDC) program correction. The analyst will receive a message on the console and must manually correct using LOBDC (35:5-12).

2.9.2.2 Manual Element Set Maintenance. The program MANDC provides the analyst manual control of the differential correction process. MANDC will directly queue the ELTRAN program to process the observations and conduct the correction process (24:204.1). This program is used to process those satellites with non-routine orbits (special-interest satellites). The processing of observations remains the same for a manually-corrected satellite.

2.10 Summary of Past SSN Assessments

Since the beginnings of the SSN there have been several assessments of sensors, network capabilities, and tasking operations. As of September 1992, assessments were still being conducted with the goal of continual process improvement (30). For the purpose of this research, two past assessments were evaluated and are summarized below.

2.10.1 Space Surveillance Program Architecture Report. This study sought to examine the workings of the SSN and examine possible shortfalls in the system. The initial assessment of the SSN indicated there were two general requirements for adequate space surveillance: accuracy and timeliness (31:3-5). Both of these requirements are based on required actions and responses for a wartime environment. The study found that if the timeliness requirement was dropped, the accuracy of the current sensors met the accuracy requirement for catalogue updates (31:3-5).

Based on the worldwide deployment of sensors for the SSN, the flow of information from the sensors to the SSC was of special interest to this study. The study specifically identified the command and control structure, the data processing capability, and the communication systems associated with the gathering of satellite observation information (31:4-1). The study concluded that the requirements of the catalogue support mission

is primarily a function of information throughput capability and sensor tasking priorities (31:3-5).

2.10.2 Final Report of the Surveillance Command and Control Study. This study was directed by the Secretary of the Air Force for Acquisition, Space, and SDI Programs in 1988. The stated purpose of this report was to baseline the current capability of the SSN to support anti-satellite (ASAT) operations. The primary interest of this report was: sensor accuracies, the tracking of single versus multiple targets, and timeliness (9:1-1). To examine these areas, five area studies were accomplished: Flash Elset generation, Orbit Prediction Accuracy, Individual Sensor Calibration, SSC Capability, and Sensor Coverage (9:1-3).

The Flash Elset study was conducted to determine the ability of a sensor to generate an accurate Flash Elset. The Flash Elset study was directly applicable to the ASAT mission. The Orbit Prediction Accuracy study was conducted to test the different propagation theories with respect to accuracy, time dependence, and amounts of sensor data. The Sensor Calibration study was conducted to verify the accuracy of the sensors in the SSN. The SSC Capability study was conducted to evaluate the operations of the SSC. The Sensor Coverage study was conducted to evaluate the locations of the sensors in the SSN relative to the ASAT mission.

Of primary interest to this research were the Sensor Calibration study, SSC Capability study, and the Orbital Prediction Accuracy study. The Sensor Calibration study provided measurement biases and standard deviations (σ) for sensor measurements. This information was input into our differential corrector and used to analyze the satellite population. The SSC Capability study provided background information into the operation of the SSC and the sensor tasking methodology. The Orbital Prediction Accuracy study provided background on the orbital prediction problem.

These area studies resulted in several conclusions (9:1-13):

- The computer usage and processing time increased as the complexity of the orbit prediction model increased.

- The computer processing time increased as the quantity of observations increased.
- Certain gaps in the sensor coverage caused noticeable delays in the processing of information.

2.10.2.1 Sensor Calibration Study. This study examined accuracy in terms of azimuth, elevation, range, and range rate measurements (9:1-10). Even though sensor calibration is relatively stable, all sensors must be calibrated on a regular basis (10:3-3). The interval of calibration varies from sensor to sensor, but the process must be routinely accomplished to maintain observation accuracy (10:2-15).

Calibration is accomplished by tasking the sensor to track a satellite with a known and stable orbit (10:2-15). These observations are then sent to the SSC and processed on calibration analysis programs. These analysis programs generate the values for bias and standard deviation (σ) (10:2-15). The updated calibration results are then distributed to the sensors via electronic message on a daily basis (2).

This study used the Navy Transit satellite as a known object for sensor calibration (10:3-1). After the readings were taken, the observations were processed through the SSC computer systems to generate biases and standard deviations. This study noted that there are multiple sources of error which will impact the values of bias and σ. Examples of these errors are: refraction, signal to noise ratio, and satellite-dependent physical characteristics (i.e., shape, orientation, and surface area) (10:3-1). However, all of these errors are accounted for in the values for bias and σ published by the SSC.

2.10.2.2 Orbital Prediction Accuracy Study. To determine the accuracy of an orbital prediction model, three studies were conducted. These studies looked at historical data, simulation results, and historic data from NAVSPASUR (9:1-7). These studies indicated errors are highest when the time span of data is short. This error was seen to level out after three hours of available tracking data (9:1-8). This study also found the quality of the prediction was directly proportional to the accuracy of the sensor[8] providing the information (9:1-9).

[8]In terms of bias and σ.

The evaluation of the orbit prediction theories indicated that special perturbation theory is always the best predictor. However, the application of this theory takes considerable computer time. General perturbation theory will provide adequate results but there is some accuracy penalty. However, the accuracy penalty was determined acceptable if five or six hours of observations were utilized (9:1-9).

2.10.2.3 SSC Capability Study. The SSC Capability study was important because of the ASAT nature of the original project. Researchers needed to determine how sensor tasking would impact the generation of elsets. This study determined that tasking was important to ensure the proper dispersion and number of observations were gathered for a satellite (10:2-14). To maintain the elsets, this study recommended orbits be determined every 1-2 days for LEO satellites and 7-10 days for high orbit satellites (11:3-9).

III. THEORETICAL BACKGROUND

3.1 Element/Covariance Propagation

This section establishes the dynamics used in the propagation portion of our dynamics model as well as the perturbations analysis of our satellite constellation. The dynamics used are based strictly on the J_2 perturbations. Section 3.1.1 shows the derivation of the state solution for the elements, and Section 3.1.2 establishes the state-transition matrix.

3.1.1 The State Solution. Since we are dealing with mean Keplerian orbital elements, we are interested in the non-periodic terms[1] in the time rate of change in the orbital elements. The terms in the expansion of the disturbing function (R) which do not include the mean anomaly, the only periodic term, are the secular terms. It has been shown the secular portion of the J_2 disturbing function is (37:83):

$$R_{2,sec} = -\frac{\mu J_2 R_\oplus^2}{2a^3\left(1-e^2\right)^{3/2}}\left(\frac{3}{2}\sin^2 i - 1\right) \tag{3.1}$$

Reference (37:47–48) provides the disturbing-function form of the Lagrange planetary equations:

$$\dot{a} = \frac{da}{dt} = \frac{2}{na}\frac{\partial R}{\partial M_0} \tag{3.2}$$

$$\dot{e} = \frac{de}{dt} = \frac{1-e^2}{na^2 e}\frac{\partial R}{\partial M_0} - \frac{\sqrt{1-e^2}}{na^2 e}\frac{\partial R}{\partial \omega} \tag{3.3}$$

$$\dot{i} = \frac{di}{dt} = \frac{\cot i}{na^2\sqrt{1-e^2}}\frac{\partial R}{\partial \omega} - \frac{1}{na^2\sqrt{1-e^2}\sin i}\frac{\partial R}{\partial \Omega} \tag{3.4}$$

$$\dot{\Omega} = \frac{d\Omega}{dt} = \frac{1}{na^2\sqrt{1-e^2}\sin i}\frac{\partial R}{\partial i} \tag{3.5}$$

$$\dot{\omega} = \frac{d\omega}{dt} = \frac{\sqrt{1-e^2}}{na^2 e}\frac{\partial R}{\partial e} - \frac{\cot i}{na^2\sqrt{1-e^2}}\frac{\partial R}{\partial i} \tag{3.6}$$

$$\dot{M}_0 = \frac{dM_0}{dt} = -\frac{2}{na}\frac{\partial R}{\partial a} - \frac{1-e^2}{na^2 e}\frac{\partial R}{\partial e} \tag{3.7}$$

[1] Also called secular.

Using the Lagrange planetary equations, we can derive expressions for the time rate of change in the orbital elements. First, we must find the time derivative of the orbital elements due to $R_{2,sec}$. The expressions for these time derivatives are listed below:

$$\dot{a} = 0 \tag{3.8}$$

$$\dot{e} = 0 \tag{3.9}$$

$$\dot{i} = 0 \tag{3.10}$$

$$\dot{\Omega} = -\frac{3\mu J_2 R_\oplus^2 \cos i}{2a^5 \left(1 - e^2\right)^2 n} \tag{3.11}$$

$$\dot{\omega} = \frac{3\mu J_2 R_\oplus^2 (3 + 5\cos 2i)}{8a^5 \left(1 - e^2\right)^2 n} \tag{3.12}$$

$$\dot{M}_0 = \frac{3\mu J_2 R_\oplus^2 (1 + 3\cos 2i)}{8a^5 \left(1 - e^2\right)^{3/2} n} \tag{3.13}$$

Strictly speaking, the NORAD two-line element sets do not contain the semi-major axis (a). Instead, the Kozai mean motion (n_κ) is used. Solving this problem involves a two-step process. First, we must eliminate the semi-major axis in favor of the mean motion (n). Second, we must express the mean motion as a function of the Kozai mean motion.

First, using the identity between n and a (Equation 3.14), we rewrite Equations 3.8–3.13 without a.

$$a = \left(\frac{\mu}{n^2}\right)^{1/3} \tag{3.14}$$

$$\dot{n} = 0 \tag{3.15}$$

$$\dot{e} = 0 \tag{3.16}$$

$$\dot{i} = 0 \tag{3.17}$$

$$\dot{\Omega} = -\frac{3J_2 R_\oplus^2 n^{7/3} \cos i}{2\mu^{2/3} \left(1 - e^2\right)^2} \tag{3.18}$$

$$\dot{\omega} = \frac{3J_2 R_\oplus^2 n^{7/3}(3 + 5\cos 2i)}{8\mu^{2/3}(1-e^2)^2} \tag{3.19}$$

$$\dot{M}_0 = \frac{3J_2 R_\oplus^2 n^{7/3}(1 + 3\cos 2i)}{8\mu^{2/3}(1-e^2)^{3/2}} \tag{3.20}$$

Notice that the orbital elements which are functions of time (Ω, ω, M_0) have time derivatives ($\dot{\Omega}, \dot{\omega}, \dot{M}_0$) which are themselves not functions of elements that change with time or with time itself. Since the Kozai mean motion is defined as in Equation 3.21, if we replace n with n_κ in Equation 3.15, we have to redefine the mean anomaly at epoch to be constant (i.e., $\dot{M}_0 = 0$).

$$n_\kappa \equiv n + \dot{M}_0 \tag{3.21}$$

Integration of Equations 3.15 3.20 with respect to time yields:

$$\begin{aligned}
n_\kappa(t) &= n_{\kappa,0} && (3.22)\\
&= n_0 + \frac{3J_2 R_\oplus^2 n_0^{7/3}(1 + 3\cos 2i_0)}{8\mu^{2/3}(1-e_0^2)^{3/2}} && (3.23)\\
e(t) &= e_0 && (3.24)\\
i(t) &= i_0 && (3.25)\\
\Omega(t) &= \Omega_0 - \frac{3J_2 R_\oplus^2 n_0^{7/3}\cos i_0}{2\mu^{2/3}(1-e_0^2)^2}(t - t_0) && (3.26)\\
\omega(t) &= \omega_0 + \frac{3J_2 R_\oplus^2 n_0^{7/3}(3 + 5\cos 2i_0)}{8\mu^{2/3}(1-e_0^2)^2}(t - t_0) && (3.27)\\
M(t) &= M_0 + n_{\kappa,0}(t - t_0) && (3.28)
\end{aligned}$$

Next, since the two-line orbital element sets contain the Kozai mean motion, we need to find n_0 in terms of $n_{\kappa,0}$. We can rewrite Equation 3.23 as:

$$n_{\kappa,0} = n_0 \left\{ 1 + \frac{3 J_2 R_\oplus^2 n_0^{4/3} (1 + 3\cos 2i_0)}{8 \mu^{2/3} \left(1 - e_0^2\right)^{3/2}} \right\} \tag{3.29}$$

or,

$$n_{\kappa,0} = n_0 \left(1 + \epsilon n_0^{4/3} \right) \tag{3.30}$$

where,

$$\epsilon \equiv \frac{3 J_2 R_\oplus^2 (1 + 3\cos 2i_0)}{8 \mu^{2/3} \left(1 - e_0^2\right)^{3/2}} \tag{3.31}$$

As long as $\epsilon n_0^{4/3}$ in Equation 3.30 is small then we can make the approximation that the n_0 inside the parentheses is nearly $n_{\kappa,0}$.

$$n_0 = \frac{n_{\kappa,0}}{1 + \epsilon n_0^{4/3}} \tag{3.32}$$

$$n_0 \approx \frac{n_{\kappa,0}}{1 + \epsilon n_{\kappa,0}^{4/3}} \tag{3.33}$$

Actually, by nesting Equation 3.32 four times and substituting $n_{\kappa,0}$ for n_0, n_0 is correct through $O((\epsilon n_0^{4/3})^4)$.

$$n_0 \approx \cfrac{n_{\kappa,0}}{\left(1 + \epsilon \cfrac{n_{\kappa,0}}{\left(1 + \epsilon \cfrac{n_{\kappa,0}}{\left(1 + \epsilon \cfrac{n_{\kappa,0}}{1 + \epsilon n_{\kappa,0}^{\frac{4}{3}}}\right)^{\frac{4}{3}}}\right)^{\frac{4}{3}}}\right)^{\frac{4}{3}}} \tag{3.34}$$

Equation 3.34 can be rewritten as a power series.

$$n_0 \approx n_{\kappa,0} \left(1 - \epsilon n_{\kappa,0}^{\frac{4}{3}} + \frac{7}{3} \epsilon^2 n_{\kappa,0}^{\frac{8}{3}} - 7 \epsilon^3 n_{\kappa,0}^{4} + \frac{1925}{81} \epsilon^4 n_{\kappa,0}^{\frac{16}{3}} \right) \simeq f(n_{\kappa,0}) \tag{3.35}$$

In order for the power series to converge, we assumed $\epsilon n_0^{4/3}$ was small. Figure 3.1 is a plot of the magnitude of ϵ in canonical[2] units with an eccentricity range of 0 to .92.[3] Because the maximum mean motion in canonical units is 1, it is accurate to say that the value of $\epsilon n_0^{4/3}$ remains small for satellites with eccentricities less than .92.

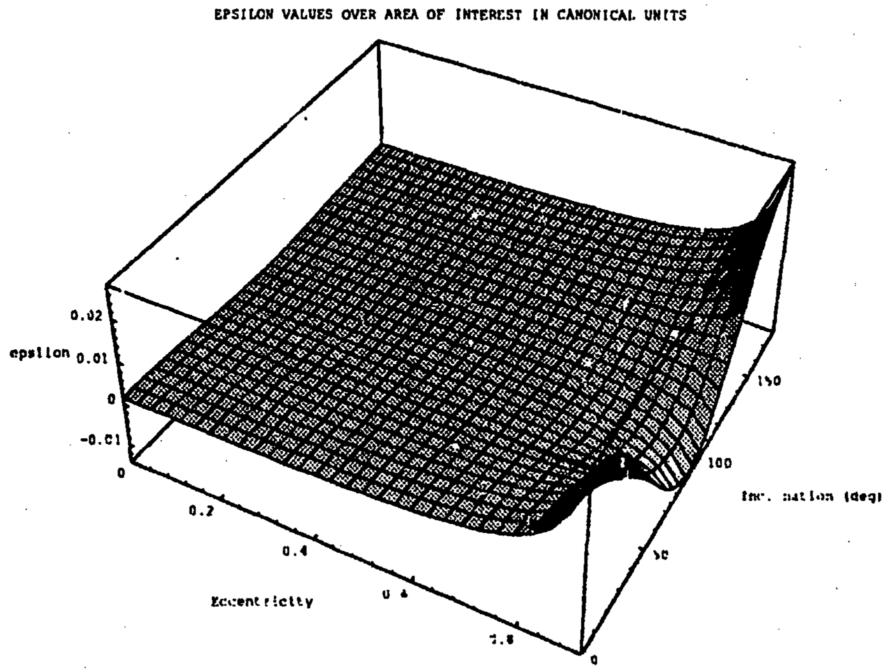


Figure 3.1. Epsilon (ϵ) Values over Area of Interest (Canonical Units).

If Equation 3.35 is substituted into Equation 3.30, we would expect the two sides to be nearly equal. To find the error induced in extracting n_0 from $n_{*,0}$, subtract $n_{*,0}$ from the result.

$$\text{error} = f(n_{*,0})\left(1 + \epsilon\left(f(n_{*,0})\right)^{4/3}\right) - n_{*,0} \tag{3.36}$$

[2] In canonical units, μ is 1 DU^3/TU^2, $R_\oplus$ is 1 DU. $R_\oplus$ is also the minimum semi-major axis thus the maximum mean motion is 1 DU/TU

[3] The maximum eccentricity of any satellite in the current population is 0.92.

Since our approximation is good to order $(\epsilon n_0^{4/3})^4$, if $\epsilon n_0^{4/3}$ is maximized then the magnitude of the error will also be at its maximum. Figure 3.2 shows the maximum error induced for all possible $n_{\kappa,0}$ and i_0 combinations.[4]

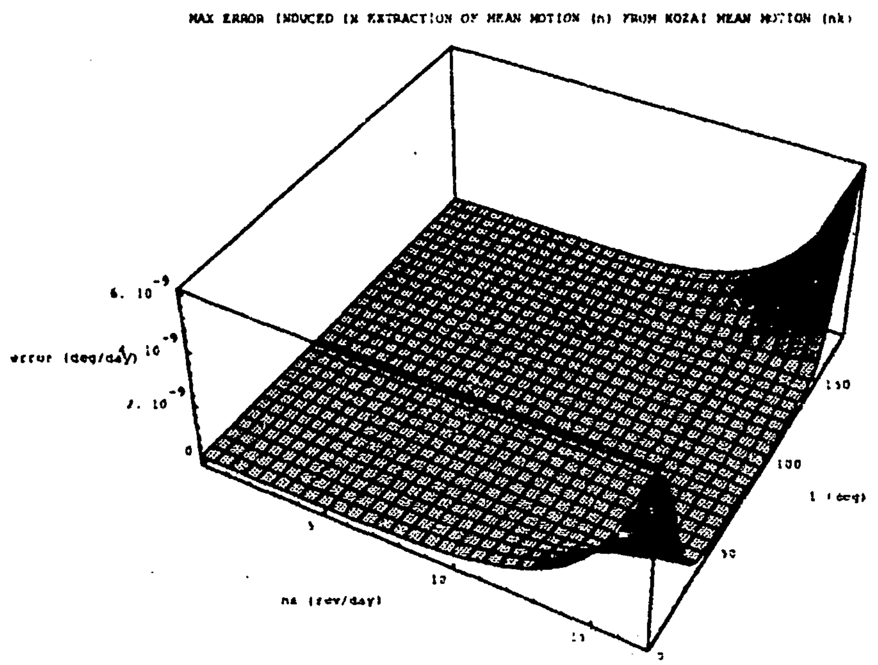


Figure 3.2. Error Induced in Extracting n_0 from $n_{\kappa,0}$.

Figure 3.2 shows clearly that the power-series expansion is very good for most mean motions and even at its worst it is less than 1×10^{-8} deg/day in error.

Summarizing the important equations so far:

$$n_\kappa(t) = n_{\kappa,0} \tag{3.37}$$

$$e(t) = e_0 \tag{3.38}$$

$$i(t) = i_0 \tag{3.39}$$

$$\Omega(t) = \Omega_0 - \frac{3J_2 R_\oplus^2 n_0^{7/3} \cos i_0}{2\mu^{2/3}(1-e_0^2)^2}(t - t_0) \tag{3.40}$$

[4] The maximum eccentricity without impacting the earth's surface was computed for each value of $n_{\kappa,0}$ and used along with i_0 in computing ϵ.

$$
\begin{aligned}
\omega(t) &= \omega_0 + \frac{3J_2R_\oplus^2 n_0^{7/3}(3 + 5\cos 2i_0)}{8\mu^{2/3}\left(1 - e_0^2\right)^2}(t - t_0) && (3.41)\\
M(t) &= M_0 + n_{\kappa,0}(t - t_0) && (3.42)
\end{aligned}
$$

where,

$$
\begin{aligned}
n_0 &\approx n_{\kappa,0}\left(1 - \epsilon n_{\kappa,0}^{\frac{4}{3}} + \frac{7}{3}\epsilon^2 n_{\kappa,0}^{\frac{8}{3}} - 7\epsilon^3 n_{\kappa,0}^{4} + \frac{1925}{81}\epsilon^4 n_{\kappa,0}^{\frac{16}{3}}\right)\\
\epsilon &\equiv \frac{3J_2R_\oplus^2(1 + 3\cos 2i_0)}{8\mu^{2/3}\left(1 - e_0^2\right)^{3/2}}
\end{aligned}
$$

Now, since we have the solution to the state variables for any time (t) given the state at some initial time (t_0), we can find the state-transition matrix analytically (by taking the partials of the state solution with respect to the initial state variables).

3.1.2 The State-Transition Matrix. The state-transition matrix is used in propagating the covariance of an estimated state. In our case, the state variables are:

$$
\mathbf{x}^T = \begin{bmatrix} n_\kappa & e & i & \Omega & \omega & M \end{bmatrix} \tag{3.43}
$$

and the state-transition matrix $\Phi(t, t_0)$ is defined as:

$$
\Phi(t, t_0) \equiv \frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}(t_0)} \tag{3.44}
$$

The partials in Φ can be written more extensively as:

$$
\Phi(t, t_0) = \begin{bmatrix}
\frac{\partial n_\kappa(t)}{\partial n_{\kappa,0}} & \frac{\partial n_\kappa(t)}{\partial e_0} & \frac{\partial n_\kappa(t)}{\partial i_0} & \frac{\partial n_\kappa(t)}{\partial \Omega_0} & \frac{\partial n_\kappa(t)}{\partial \omega_0} & \frac{\partial n_\kappa(t)}{\partial M_0}\\
\frac{\partial e(t)}{\partial n_{\kappa,0}} & \frac{\partial e(t)}{\partial e_0} & \frac{\partial e(t)}{\partial i_0} & \frac{\partial e(t)}{\partial \Omega_0} & \frac{\partial e(t)}{\partial \omega_0} & \frac{\partial e(t)}{\partial M_0}\\
\frac{\partial i(t)}{\partial n_{\kappa,0}} & \frac{\partial i(t)}{\partial e_0} & \frac{\partial i(t)}{\partial i_0} & \frac{\partial i(t)}{\partial \Omega_0} & \frac{\partial i(t)}{\partial \omega_0} & \frac{\partial i(t)}{\partial M_0}\\
\frac{\partial \Omega(t)}{\partial n_{\kappa,0}} & \frac{\partial \Omega(t)}{\partial e_0} & \frac{\partial \Omega(t)}{\partial i_0} & \frac{\partial \Omega(t)}{\partial \Omega_0} & \frac{\partial \Omega(t)}{\partial \omega_0} & \frac{\partial \Omega(t)}{\partial M_0}\\
\frac{\partial \omega(t)}{\partial n_{\kappa,0}} & \frac{\partial \omega(t)}{\partial e_0} & \frac{\partial \omega(t)}{\partial i_0} & \frac{\partial \omega(t)}{\partial \Omega_0} & \frac{\partial \omega(t)}{\partial \omega_0} & \frac{\partial \omega(t)}{\partial M_0}\\
\frac{\partial M(t)}{\partial n_{\kappa,0}} & \frac{\partial M(t)}{\partial e_0} & \frac{\partial M(t)}{\partial i_0} & \frac{\partial M(t)}{\partial \Omega_0} & \frac{\partial M(t)}{\partial \omega_0} & \frac{\partial M(t)}{\partial M_0}
\end{bmatrix} \tag{3.45}
$$

Eliminating the easy ones:

$$\Phi(t,t_0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{\partial\Omega(t)}{\partial n_{\kappa,0}} & \frac{\partial\Omega(t)}{\partial e_0} & \frac{\partial\Omega(t)}{\partial i_0} & 1 & 0 & 0 \\ \frac{\partial\omega(t)}{\partial n_{\kappa,0}} & \frac{\partial\omega(t)}{\partial e_0} & \frac{\partial\omega(t)}{\partial i_0} & 0 & 1 & 0 \\ t-t_0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.46}$$

In order to do the remaining six partials in Equation 3.46 we need the derivatives of ϵ and n_0.

$$\frac{d\epsilon}{di_0} = -\frac{9J_2R_{\oplus}^2(\sin 2i_0)}{4\mu^{2/3}(1-e_0^2)^{3/2}} \tag{3.47}$$

$$\frac{d\epsilon}{de_0} = \frac{9J_2R_{\oplus}^2(1+3\cos 2i_0)}{8\mu^{2/3}(1-e_0^2)^{5/2}}e_0 \tag{3.48}$$

$$\frac{dn_0}{dn_{\kappa,0}} = 1-\frac{7}{3}\epsilon n_{\kappa,0}^{\frac{4}{3}}+\frac{77}{9}\epsilon^2 n_{\kappa,0}^{\frac{8}{3}}-35\epsilon^3 n_{\kappa,0}^{4}+\frac{36575}{243}\epsilon^4 n_{\kappa,0}^{\frac{16}{3}} \tag{3.49}$$

$$\frac{dn_0}{d\epsilon} = n_{\kappa,0}\left(-n_{\kappa,0}^{\frac{4}{3}}+\frac{14}{3}\epsilon n_{\kappa,0}^{\frac{8}{3}}-21\epsilon^2 n_{\kappa,0}^{4}+\frac{7700}{81}\epsilon^3 n_{\kappa,0}^{\frac{16}{3}}\right) \tag{3.50}$$

The six partials are:

$$\frac{\partial\Omega(t)}{\partial n_{\kappa,0}} = -\frac{7J_2R_{\oplus}^2 n_0^{\frac{4}{3}}\cos i_0}{2\mu^{\frac{2}{3}}(1-e_0^2)^2}\cdot\frac{dn_0}{dn_{\kappa,0}}\cdot(t-t_0) \tag{3.51}$$

$$\frac{\partial\Omega(t)}{\partial e_0} = \left(-\frac{6e_0J_2R_{\oplus}^2 n_0^{\frac{7}{3}}\cos i_0}{\mu^{\frac{2}{3}}(1-e_0^2)^3}-\frac{7J_2R_{\oplus}^2 n_0^{\frac{4}{3}}\cos i_0}{2\mu^{\frac{2}{3}}(1-e_0^2)^2}\cdot\frac{dn_0}{d\epsilon}\cdot\frac{d\epsilon}{de_0}\right)\cdot(t-t_0) \tag{3.52}$$

$$\frac{\partial\Omega(t)}{\partial i_0} = \left(\frac{3J_2R_{\oplus}^2 n_0^{\frac{7}{3}}\sin i_0}{2\mu^{\frac{2}{3}}(1-e_0^2)^2}-\frac{7J_2R_{\oplus}^2 n_0^{\frac{4}{3}}\cos i_0}{2\mu^{\frac{2}{3}}(1-e_0^2)^2}\cdot\frac{dn_0}{d\epsilon}\cdot\frac{d\epsilon}{di_0}\right)\cdot(t-t_0) \tag{3.53}$$

$$\frac{\partial\omega(t)}{\partial n_{\kappa,0}} = \frac{7J_2R_{\oplus}^2 n_0^{\frac{4}{3}}(3+5\cos 2i_0)}{8\mu^{\frac{2}{3}}(1-e_0^2)^2}\cdot\frac{dn_0}{dn_{\kappa,0}}\cdot(t-t_0) \tag{3.54}$$

$$\frac{\partial\omega(t)}{\partial e_0} = \left(\frac{3e_0J_2R_{\oplus}^2 n_0^{\frac{7}{3}}(3+5\cos 2i_0)}{2\mu^{\frac{2}{3}}(1-e_0^2)^3}+\frac{7J_2R_{\oplus}^2 n_0^{\frac{4}{3}}(3+5\cos 2i_0}{8\mu^{\frac{2}{3}}(1-e_0^2)^2}\cdot\frac{dn_0}{d\epsilon}\cdot\frac{d\epsilon}{de_0}\right)\cdot(t-t_0)$$

$$(3.55)$$

$$\frac{\partial \omega(t)}{\partial i_0} = \left(-\frac{15 J_2 R_\oplus^2 n_0^{\frac{7}{3}} \sin 2i_0}{4\mu^{\frac{2}{3}} (1-e_0^2)^2} + \frac{7 J_2 R_\oplus^2 n_0^{\frac{4}{3}} (3 + 5\cos 2i_0)}{8\mu^{\frac{2}{3}} (1-e_0^2)^2} \cdot \frac{dn_0}{d\epsilon} \cdot \frac{d\epsilon}{di_0} \right) \cdot (t - t_0) \quad (3.56)$$

In Section 3.1.1, we derived the state solution as a function of time. We used the state solution equations to derive an analytic Φ. We will use Φ, provided in Equations 3.46 through 3.56, for propagating our state-covariance.

3.2 *Differential Correction*

This section reviews the differential correction theory used in our model. The estimation algorithm we used in this research is a sequential estimator, specifically a Bayes filter algorithm, as presented in (38:55–94). The Bayes filter is basically a weighted least squares filter modified to use the previous state estimate and the covariance of that estimate as the first data input to the filter. Before discussing the Bayes filter, a review of least squares estimation is required.

3.2.1 Least Squares Estimation. Least squares is the basis for estimation theory. Least squares estimates the state of a dynamic system at some time t_0 ($\mathbf{x}(t_0)$) by using a series of actual measurements, or observations ($\hat{\mathbf{O}}(t_i)$) taken at different times t_i. In orbit determination, least squares attempts to determine the orbit most likely to produce the observational data. The estimate is determined by minimizing the sum of the squares of the residuals between the actual observations ($\hat{\mathbf{O}}(t_i)$) and predicted observations ($\mathbf{O}(t_i)$) based upon a reference state. Weighted least squares comes from associating an observation weighting, or instrument covariance matrix ($Q(t_i)$), with each observation vector $\hat{\mathbf{O}}(t_i)$.

A set of observation data at different times t_i which is linearly related to the system state at that time can be written as

$$\hat{\mathbf{O}}(t_i) = H_i \, \mathbf{x}(t_i) + \mathbf{e}_i \quad (3.57)$$

where $\mathbf{e}_i$ is the true error in the data and H_i is the linear relationship between the observations and the state at time t_i ($\mathbf{x}(t_i)$).

In a linear dynamical system, the state at time t_i can be written as a function of the state at time t_0 as shown in Equation 3.58 below.

$$\mathbf{x}(t_i) = \Phi(t_i, t_0)\,\mathbf{x}(t_0) \tag{3.58}$$

Solving Equation 3.57 for $\mathbf{e}_i$ and expressing it in terms of state at time t_0 gives the following:

$$\begin{aligned} \mathbf{e}_i &= \hat{\mathbf{O}}(t_i) - H_i\,\Phi(t_i, t_0)\,\mathbf{x}(t_0) && (3.59)\\ &= \hat{\mathbf{O}}(t_i) - T_i\,\mathbf{x}(t_0) && (3.60) \end{aligned}$$

where the observation matrix, T_i, as defined in Equation 3.61, relates the observations at time t_i to the state vector at the epoch time, t_0.

$$T_i \equiv H_i\,\Phi(t_i, t_0) \tag{3.61}$$

Using a matrix shorthand notation, we can define the total observation data vector, $\hat{\mathbf{O}}$, the total observation matrix, T, and the total instrument covariance matrix Q, to be as shown in Equations 3.62, 3.63, and 3.64.

$$\hat{\mathbf{O}} \equiv \begin{bmatrix} \hat{\mathbf{O}}(t_1) \\ \hat{\mathbf{O}}(t_2) \\ \vdots \\ \hat{\mathbf{O}}(t_N) \end{bmatrix} \tag{3.62}$$

$$T \equiv \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_N \end{bmatrix} = \begin{bmatrix} H_1\,\Phi(t_1, t_0) \\ H_2\,\Phi(t_2, t_0) \\ \vdots \\ H_N\,\Phi(t_N, t_0) \end{bmatrix} \tag{3.63}$$

$$Q \equiv \begin{bmatrix} Q_1 & 0 & \cdots & 0 \\ 0 & Q_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q_N \end{bmatrix} \tag{3.64}$$

where N is the total number of independent observations, $\hat{\mathbf{O}}(t_i)$.

Replacing the true error $\mathbf{e}$ and the true state $\mathbf{x}(t_0)$ with the residual $\mathbf{r}$ and the estimated state $\bar{\mathbf{x}}(t_0)$, Equation 3.60 can be restated as

$$\mathbf{r} = \hat{\mathbf{O}} - T\,\bar{\mathbf{x}}(t_0) \tag{3.65}$$

The probability of obtaining these particular residuals is then shown by Equation 3.66.

$$\mathrm{P}(\mathbf{r}) = (2\pi)^{-\frac{N}{2}}\ |Q|\ \mathrm{e}^{(-\frac{1}{2}\mathbf{r}^T Q^{-1}\mathbf{r})} \tag{3.66}$$

To determine the state estimate $\bar{\mathbf{x}}$ which maximizes $\mathrm{P}(\mathbf{r})$, the quantity $(\mathbf{r}^T Q^{-1}\mathbf{r})$ must be minimized. This results in the minimization problem

$$\frac{\partial}{\partial \bar{\mathbf{x}}}\left((\hat{\mathbf{O}} - T\bar{\mathbf{x}})^T Q^{-1}(\hat{\mathbf{O}} - T\bar{\mathbf{x}})\right) = 0 \tag{3.67}$$

which, when solved for the estimated state, $\bar{\mathbf{x}}(t_0)$, gives

$$\bar{\mathbf{x}}(t_0) = (\,T^T\,Q^{-1}\,T\,)^{-1}\,T^T\,Q^{-1}\,\hat{\mathbf{O}} \tag{3.68}$$

where the quantity $(T^T Q^{-1} T)^{-1}$ is the covariance of the estimated state, $P_{\bar{\mathbf{x}}}$.

In orbit determination, the dynamics are very non-linear. However, if the difference between the true state and a reference state $(\delta\mathbf{x})$ is small, the non-linear dynamics may be linearized about this reference state, $\mathbf{x}^*$. In addition, the observation data is also a nonlinear function of the time, t_i, and the state at that time, $\mathbf{x}(t_i)$. This observation relationship can be written as shown in Equation 3.69.

$$\mathbf{O}(t_i) = G(\mathbf{x}(t_i), t_i) \tag{3.69}$$

This relationship can also be linearized through the use of a Taylor series expansion and written in the form shown in Equation 3.70

$$\mathbf{e}_i \approx \frac{\partial \mathbf{G}}{\partial \mathbf{x}} \delta \mathbf{x}(t_i) = H_i \, \delta \mathbf{x}(t_i) \tag{3.70}$$

where $\mathbf{e}_i$ is the true error in the actual observation data and H_i is the data linearization matrix for the observations at time t_i.

The residuals between actual observation data and calculated observations based upon the reference state are then calculated as shown in Equation 3.71.

$$\mathbf{r}_i = \hat{\mathbf{O}}(t_i) - \mathbf{G}(\mathbf{x}(t_i), t_i) \tag{3.71}$$

The state transition matrix, $\Phi(t, t_0)$ relates changes to the state at time t_i to changes at time t_0 as shown in Equation 3.72.

$$\delta \mathbf{x}(t_i) = \Phi(t_i, t_0) \, \delta \mathbf{x}(t_0) \tag{3.72}$$

The residual vector $\mathbf{r}_i$ is then written as shown in Equation 3.73.

$$\mathbf{r}_i \approx H_i \, \delta \mathbf{x}(t_i) = H_i \, \Phi(t_i, t_0) \, \delta \mathbf{x}(t_0) = T_i \, \delta \mathbf{x}(t_0) \tag{3.73}$$

Using the same method that arrived at Equation 3.68 for linear least squares, the correction to the estimated state can be determined as shown in Equation 3.74.

$$\delta \mathbf{x}(t_0) = (T^T Q^{-1} T)^{-1} T^T Q^{-1} \mathbf{r} \tag{3.74}$$

with the covariance of the estimated state being as shown in Equation 3.75.

$$P_{\hat{\mathbf{x}}} = (T^T Q^{-1} T)^{-1} \tag{3.75}$$

The new estimate of the state is determined as shown in Equation 3.76.

$$\bar{\mathbf{x}}(t_0) = \mathbf{x}^*(t_0) + \delta\mathbf{x}(t_0) \tag{3.76}$$

3.2.2 Bayes Estimation Algorithm. The Bayes filter uses the old estimate of the state, $\bar{\mathbf{x}}_{old}$, and the state covariance of the old estimate, P_{old} as the initial data point along with a new set of observations ($\hat{\mathbf{O}}$) and their associated covariance (Q_{new}).

Equations 3.62 through 3.64 as presented in Section 3.2.1 above are then rewritten as

$$T = \begin{bmatrix} I \\ T_{new} \end{bmatrix} \tag{3.77}$$

$$Q = \begin{bmatrix} P_{old} & 0 \\ 0 & Q_{new} \end{bmatrix} \tag{3.78}$$

$$\mathbf{r} = \begin{bmatrix} \bar{\mathbf{x}}_{old} - \mathbf{x}^* \\ \hat{\mathbf{O}} - \mathrm{G}(\mathbf{x}(t_i), t_i) \end{bmatrix} \tag{3.79}$$

The inverse of the new covariance, P_{new}^{-1} is then calculated as shown in Equation 3.81.

$$P_{new}^{-1} = (T^T Q^{-1} T)^{-1} \tag{3.80}$$

$$= P_{old}^{-1} + (T_{new}^T Q_{new}^{-1} T_{new})^{-1} \tag{3.81}$$

Once the new covariance is calculated, the correction to the state, $\delta\mathbf{x}$ is then calculated as shown in Equation 3.83.

$$\delta\mathbf{x}(t_0) = P_{new}\, T^T\, Q^{-1}\, \mathbf{r} \tag{3.82}$$

$$= P_{new}\, (\, P_{old}^{-1}\, (\bar{\mathbf{x}}_{old} - \mathbf{x}^*) + T_{new}^T Q_{new}^{-1} \mathbf{r}_{new}\,) \tag{3.83}$$

If the reference state ($\mathbf{x}^*$) has converged to a solution, the new estimate of the state is as shown in Equation 3.84.

$$\bar{\mathbf{x}}(t_0) = \mathbf{x}^*(t_0) + \delta\mathbf{x}(t_0) \tag{3.84}$$

The entire sequential estimation process can be summarized using Equations 3.81, 3.83, and 3.84. Given the prior estimate of the state and the state covariance ($\bar{\mathbf{x}}_{old}$ and P_{old}) and a set of new observations ($\hat{\mathbf{O}}$) and associated covariance (Q), a new estimate can be determined.

3.3 Numerical Calculation of the Observation Matrix.

As seen in Section 3.2, the observation matrix is calculated using Equation 3.85.

$$T_i = H_i\,\Phi(t_i, t_0) \tag{3.85}$$

This is an $m \times n$ matrix where m is the number of observation measurements and n is the number of state elements. It relates the observations at time t_i to the state vector at the epoch time, t_0. As discussed in Section 3.1, simplifications were made in the algorithm used to determine the state transition matrix ($\Phi(t_i, t_0)$). Because of these simplifications, we did not want to use Equation 3.85 to calculate T_i. Since the NORAD orbital model, SGP4 was available, a numerical method for calculating this matrix was used.

Given an observation consisting of four measurements and a state consisting of six elements, an expansion of Equation 3.85 becomes:

$$T_i = \begin{bmatrix} \frac{\partial O_1(t_i)}{\partial x_1(t_0)} & \frac{\partial O_1(t_i)}{\partial x_2(t_0)} & \frac{\partial O_1(t_i)}{\partial x_3(t_0)} & \frac{\partial O_1(t_i)}{\partial x_4(t_0)} & \frac{\partial O_1(t_i)}{\partial x_5(t_0)} & \frac{\partial O_1(t_i)}{\partial x_6(t_0)} \\ \frac{\partial O_2(t_i)}{\partial x_1(t_0)} & \frac{\partial O_2(t_i)}{\partial x_2(t_0)} & \frac{\partial O_2(t_i)}{\partial x_3(t_0)} & \frac{\partial O_2(t_i)}{\partial x_4(t_0)} & \frac{\partial O_2(t_i)}{\partial x_5(t_0)} & \frac{\partial O_2(t_i)}{\partial x_6(t_0)} \\ \frac{\partial O_3(t_i)}{\partial x_1(t_0)} & \frac{\partial O_3(t_i)}{\partial x_2(t_0)} & \frac{\partial O_3(t_i)}{\partial x_3(t_0)} & \frac{\partial O_3(t_i)}{\partial x_4(t_0)} & \frac{\partial O_3(t_i)}{\partial x_5(t_0)} & \frac{\partial O_3(t_i)}{\partial x_6(t_0)} \\ \frac{\partial O_4(t_i)}{\partial x_1(t_0)} & \frac{\partial O_4(t_i)}{\partial x_2(t_0)} & \frac{\partial O_4(t_i)}{\partial x_3(t_0)} & \frac{\partial O_4(t_i)}{\partial x_4(t_0)} & \frac{\partial O_4(t_i)}{\partial x_5(t_0)} & \frac{\partial O_4(t_i)}{\partial x_6(t_0)} \end{bmatrix} \tag{3.86}$$

The elements of this matrix could be calculated numerically by using the definition of a derivative. The definition of a derivative results in the evaluation of the $\{i,j\}$ element of T_i as shown in Equation 3.87.

$$\frac{\partial O_j(t_i)}{\partial x_k(t_0)} = \lim_{\Delta x_k \to 0} \frac{O_j(\mathbf{x}(t_0) + \Delta x_k(t_0), t_i) - O_j(\mathbf{x}(t_0), t_i)}{\Delta x_k(t_0)} \tag{3.87}$$

If Equation 3.87 is evaluated, not as $\lim_{\Delta \mathbf{x}_k \to 0}$, but as $\Delta \mathbf{x}_k$ approaches an incrementally small value, the elements of the observation matrix, T_i, can be expressed as:

$$\frac{\partial O_j(t_i)}{\partial x_k(t_0)} \cong \frac{O_j(\mathbf{x}(t_0) + \Delta x_k(t_0), t_i) - O_j(\mathbf{x}(t_0), t_i)}{\Delta x_k(t_0)} \tag{3.88}$$

3.4 Variance Reduction Techniques

We considered two variance reduction techniques available for our use. The first technique uses common random numbers (CRNs). The second technique uses antithetic variates which are random numbers that come in pairs such that they cancel out their effects when averaged. For an excellent source on the subject of variance reduction see (21:Chapter 11). The following is a brief overview of these two techniques.

3.4.1 Common Random Numbers (CRNs). Common random numbers is a variance reduction technique applicable to experimental testing of different system configurations. The general principle behind the application of CRNs is that differences in performance between alternate combinations, or system configurations, of an experiment should be due to differences only in the system configurations and not in the experimental conditions. Therefore, the different system configurations should be tested under as close to identical experimental conditions as possible. In simulation, the experimental conditions are the random variates generated for use in the model (21:613–614).

The ideal use of CRN is to use the same random number, or random number stream, for exactly the same purpose in one system configurations test as another. This is called "synchronization." This synchronization gives you a high confidence that the difference in performance between different system configurations is due only to the different configurations, and has no contribution inserted due to different experimental conditions (generated random variates) (21:617).

3.4.2 Antithetic Variates (AVs). Antithetic variates is a variance reduction technique applicable to a single system configuration. The general principle of AVs is to perform experimental test on the system in pairs such that an observation on one run is negatively correlated to an observatio[illegible]e other run. If the two observations are then

averaged and the single data point used in the analysis, that data point should be closer to the expected value of that observation than if the two observations were independent (21:628–629).

AV attempts to include a negative correlation by using complementary pairs of random numbers. For example, if the first random number of a pair was a uniform 0 to 1 random number (U_k), the second number in the pair, its complement, would become $1 - U_k$ where the subscript k represents a particular uniform random number observation. The fundamental requirement that a model should satisfy for AVs to work is that its response to a random number used for a particular purpose be monotonic, in either direction. As in CRN, synchronization must also be applied (21:629–630).

IV. METHODOLOGY

The basic methodology we used in this research and analysis consisted of a four-step process:

1. Perform an analysis of the earth's satellite population in order to select a representative sample of satellites for analysis.
2. Develop a "truth model" to produce highly accurate, simulated SSN sensor observations of the sample satellites.
3. Develop a differential corrector to correct the satellites orbital elements, and calculate the vector magnitude of the error between the "truth" position and the position predicted by the USSPACECOM/NORAD orbital model. Run this model for each of the representative satellites using different observations rates and correction intervals.
4. Perform a statistical analysis on this data trying to determine a steady-state position error for different observation rate correction interval combinations.

4.1 Analysis of Earth Satellite Population

There are two primary reasons for analyzing the satellite population. The first reason is to characterize the composition of earth orbiting objects as completely as possible. Prior to undertaking any analysis of earth orbiting satellites, an understanding of their characteristics and composition is essential. The second reason for performing this analysis is to be able to develop an initial classification scheme to allow the selection of a representative sample of satellites for further analysis as part of this research effort.

4.1.1 Satellite Composition Analysis. We chose to examine the composition of the most current NORAD two-line element set file available to us. This was the March 1990 two line element set file (1990.TLE). To do this, a FORTRAN program, REDUCE_TLE.FOR, was written to select the first occurrence of each satellite's two line element set from this file and to output it into a Mathematica readable file OE9003.FIX. The OE9003.FIX file contains this first occurrence of each satellite in the 1990.TLE file unless it does not pass a

series of simple tests as summarized in Appendix F.3.[1] Each entry into the OE9003.FIX file contains the epoch, a counter number, the NORAD catalog number, and the nine orbital elements from the NORAD two-line element set: Kozai mean motion (n_k), the first time derivative of mean motion divided by two ($\dot{n}/2$), the second time derivative of mean motion divided by six ($\ddot{n}/6$), inclination (i_0), node (Ω_0), argument of perigee (ω_0), eccentricity (e_0), bstar (B^*), and finally the mean anomaly at epoch (M_0).

Once in Mathematica-readable format, we processed graphs of each of the above elements and variables that are derivable from the ones provided, such as the true anomaly. The main purpose of the graphs was to allow easy examination of the distributions of each variable. We were especially interested in two areas: correlations between any two orbital elements and how the satellite population fell into the various Gabbard classes. On occasion we generated plots showing the relationships between three variables. In particular, we used 3 variable plots to determine if there is a relation between B^* and the $\dot{n}/2$ and $\ddot{n}/6$ terms. However, the problem with this investigation was attempting to see a three-dimensional space which was limited to a two-dimensional sheet of paper.

4.1.2 Perturbations Analysis. Since the J_2 perturbations are the primary source of perturbing effects on most earth orbiting satellites, this was the only perturbation effect analyzed. We wanted to identify the magnitude of this perturbation effect on the satellites in the current population. As shown in Section 3.1.1, the only three orbital elements affected by J_2 perturbations are the mean anomaly, argument of perigee, and ascending node. The time derivatives of these three elements elements were plotted as a function of the elements that affect the magnitude of the perturbations (mean motion, eccentricity, and inclination). These plots allowed us to visualize the relative perturbing effects on the satellites in the current population.

In addition, a propagation routine for the differential corrector model was required. This routine had to perform two functions. First, determine the orbital elements after some propagation time given the elements prior to propagation. Second, determine the state covariance matrix after a given propagation time given the state covariance prior to

[1] The satellites in the OE9003.FIX file are henceforth referred to as the "Current Population."

propagation. As shown in Section 3.1.1, after deriving the secular time-rate-of-change of each orbital element, it was possible to analytically integrate the equations with respect to time. This allows the expression of the orbital elements as a function of time. By differentiating these expressions with respect to the orbital elements, it is possible to find the state transition matrix as a function of time.

Had this not been possible, we would have been forced to perform a numerical integration routine to find the elements and state-transition matrix after some propagation time. We decided that an analytic approach was preferable over a numeric approach due to the computational time involved.

4.1.3 Classification Scheme Development. An orbital-element-based classification scheme was developed based on both the analysis of the current composition of satellites and the analysis of the effect of J_2 perturbations. The classes were based strictly on only the three orbital elements that influence the magnitude of the J_2 perturbations. The code that performed the actual separation of all satellites in the current population into their respective classes was the same code that decoded the 1990.TLE file.

A basic assumption made in the development of this classification scheme is that that satellites with similar perturbation effects would have similar observation requirements for maintaining the orbital element sets.

4.1.4 Representative Satellite Selection. After the satellite population was divided into their various classes, we selected a representative satellite from each class. The satellite closest to the "average" satellite was selected. To determine which satellite was closest to the average, the mean and variance of each class was calculated for the relevant orbital elements. Then, the deviation from the mean of each orbital element was weighted by the variance of the element (smaller variances having larger significance). We selected the satellite with the smallest overall deviation as our representative satellite. In other words, to make our selection we formed an *n* dimensional Gaussian distribution, and determined which satellite occurred closest to the mean.

4.2 SGP Program Library

In an attempt to simulate the methods used by USSPACECOM as closely as possible, we used the SGP4 and SDP4 orbital models in use by USSPACECOM. The specific models used throughout this research come from the SGP program library. This library contains a set of Turbo Pascal[2] 6.0 units and documentation written by Dr. Thomas S. Kelso (19). The main components of this library are the SGP4 and SDP4 orbital models which were converted from NORAD FORTRAN code as published in Reference (16). The remaining units contain routines that are either necessary for SGP4's operation or aid in interfacing SGP4 with both the truth model programs and the dynamics model programs. The library consists of the following units:

- *SGP4SDP4*. Full implementation of NORAD SGP4 and SDP4 models.
- *SGP_INIT*. Contains constants, variables, and type declarations needed to initialize *SGP4SDP4*.
- *SGP_INTF*. Constants, variables, and type declarations needed to interface between *SGP4SDP4* and *SGP_CONV* (and other programs).
- *SGP_MATH*. Various trigonometric and mathematical routines.
- *SGP_TIME*. Time based routines for converting among time systems.
- *SUPPORT*. General support routines for machine dependent features.
- *SGP_CONV*. Routines for converting two line data and SGP4 state vectors.
- *SGP_IN*. Routines to simplify input of data (with error checking).
- *SGP_OUT*. Routines to output program results in standard formats.
- *SGP_OBS*. Observer dependent routines for calculating topocentric information.
- *MINMAX*. Minimum/maximum functions.
- *SOLAR*. Routines for calculating the position of the sun and if a satellite is in earth umbral eclipse.

[2] Turbo Pascal is a trademark of Borland International

A copy of the complete SGP Library Documentation is located in Appendix A. The source code listings of all SGP Library routines used in this research are in Appendix B.

4.3 Truth Model

The purpose of the "truth model" is to produce highly accurate "truth" observations. These observations are used to simulate the observations gathered by sensors of the SSN for a specific earth-orbiting satellite. To allow a Monte Carlo analysis, the model is capable of producing multiple random sets of observations at various specified observations rates. The methodology used in the selection of the number of random sets and observation rates is presented in Section 4.5.

4.3.1 Model Development. The primary element required for this truth model is a highly accurate orbital propagator. A commercially available orbital model, the High Precision Orbit Propagator (HPOP) program, Version 1.1, produced by Microcosm, Inc. of Torrance, CA was used for our research. HPOP uses the Runge-Kutta-Fehlberg method of order 7(8) to integrate the equations of motion. It also includes accurate models for all of the major perturbations affecting an earth-orbiting satellite and all major predictable motions of the earth that affect a satellite's apparent position (15:1 2). These perturbation models include:

- Geopotential effects. Uses an unclassified 21 × 21 spherical harmonic expansion model (Goddard Earth Model 10B) of the earth geopotential.
- Solar and lunar gravitational effects. Assumes point masses and uses the U.S. Naval Observatory compressed ephemeris to predict the positions of the sun and moon.
- Atmospheric drag. Assumes single-collision specular reflection and uses the Harris-Priester atmospheric model, modified to take into account the diurnal bulge, to compute the atmospheric density.
- Solar radiation pressure.
- Precession of the equinox.
- Nutation.

- Diurnal rotation.

- Barycentric displacement.

The HPOP model produces highly accurate earth centered inertial (ECI) position and velocity vectors, at specified time intervals, over a specified length of time, for a given input ECI state vector. However, our research uses a satellite orbital element set, not an ECI state vector, as the initial data point. Additionally, our research needs the model output expressed as sensor observation data (azimuth or right ascension, elevation or declination, range, and range rate), not ECI state vectors. Three Turbo Pascal interface programs, HPOP_IN, CONVERT, and RSELECT, written by Dr. Thomas S. Kelso, were provided to translate HPOP data into the formats required for our research.

HPOP_IN uses SGP Library routines, primarily the procedures of SGP4 and SDP4 in the unit *SGP4SDP4*, to convert satellite orbital element set data into an ECI position and velocity state vector. It places this data, along with the specified interval for propagation and the propagation step size, into the input format required for HPOP.

CONVERT converts each of the HPOP "truth" ECI position and velocity vectors into SSN sensor observations (e.g., time, azimuth, elevation, range, range rate) given the SSN sensor locations, observational limitations, and data types. For our research, the SSN locations, limitations, and data types used are listed in Tables 2.2, 2.3, and 2.15 respectively. CONVERT uses the SGP library procedure *Calculate_Obs* or *Calculate_RADec* from the unit *SGP_OBS*, to perform the ECI to topocentric coordinate transformation of the ECI position and velocity vectors to either azimuth, elevation range and range rate observations or right ascension and declination observations. Additionally, CONVERT incorporates the effects of atmospheric refraction on the different types of observations.

RSELECT uses the observation file created by CONVERT and performs three functions: indexes the file by day of observation, adds random noise to the individual observations, and generates random observation files based on the required maximum number of observations per day and random observation sets.

To make the random selection, RSELECT first indexes the observation file to separate the observations by day. Based on the required number of observations per day, RSELECT

then creates the files by randomly selecting observations based on the required *maximum* number of observations per day.

To reduce the variance in the observations, these observation sets are inclusive. For example, the observations which would be included in the 2 observations-per-day set are included in the 4, 6, 8 and 10 observations-per-day set. Likewise, the two additional observations used to make the 4 observations-per-day set from the 2 observations-per-day set are used in the 6, 8, and 10 observations-per-day sets. This process is continued until the last two random observations chosen are added to only the 10 observations-per-day set.

Based on the required number of random sets, RSELECT uses a different random number seed to select observations from the indexed days. The seed numbers were created using the SLAM simulation software (28). These seed numbers were then hard coded into the Turbo Pascal unit *Gauss2*. This unit uses a numerical procedure to create a Gaussian random number with a mean of zero based on the input seed number (27:213 237). This random number process is located in a function called *GRandom* and used by RSELECT. The purpose of this random number process is twofold: specific seed numbers allow for repetition of results while still inducing a random process into the observation processing methodology.

In addition to the observation conversion and selection process performed in CONVERT and RSELECT, these programs also pass, unchanged, the ECI position vector produced by HPOP. This vector is used within the differential corrector to calculate the vector magnitude of the true position error (VMAGT).

When combined, these four truth model programs take a satellite orbital element set and create from it a specified number of sets of random observations at various specified observation rates. The flow of the truth model is shown in Figure 4.1.

4.3.2 Model Execution. The elements of the truth model a executed to flow as shown in Figure 4.1. A description of the execution method and input/output file formats for each of the elements of the truth model follows.

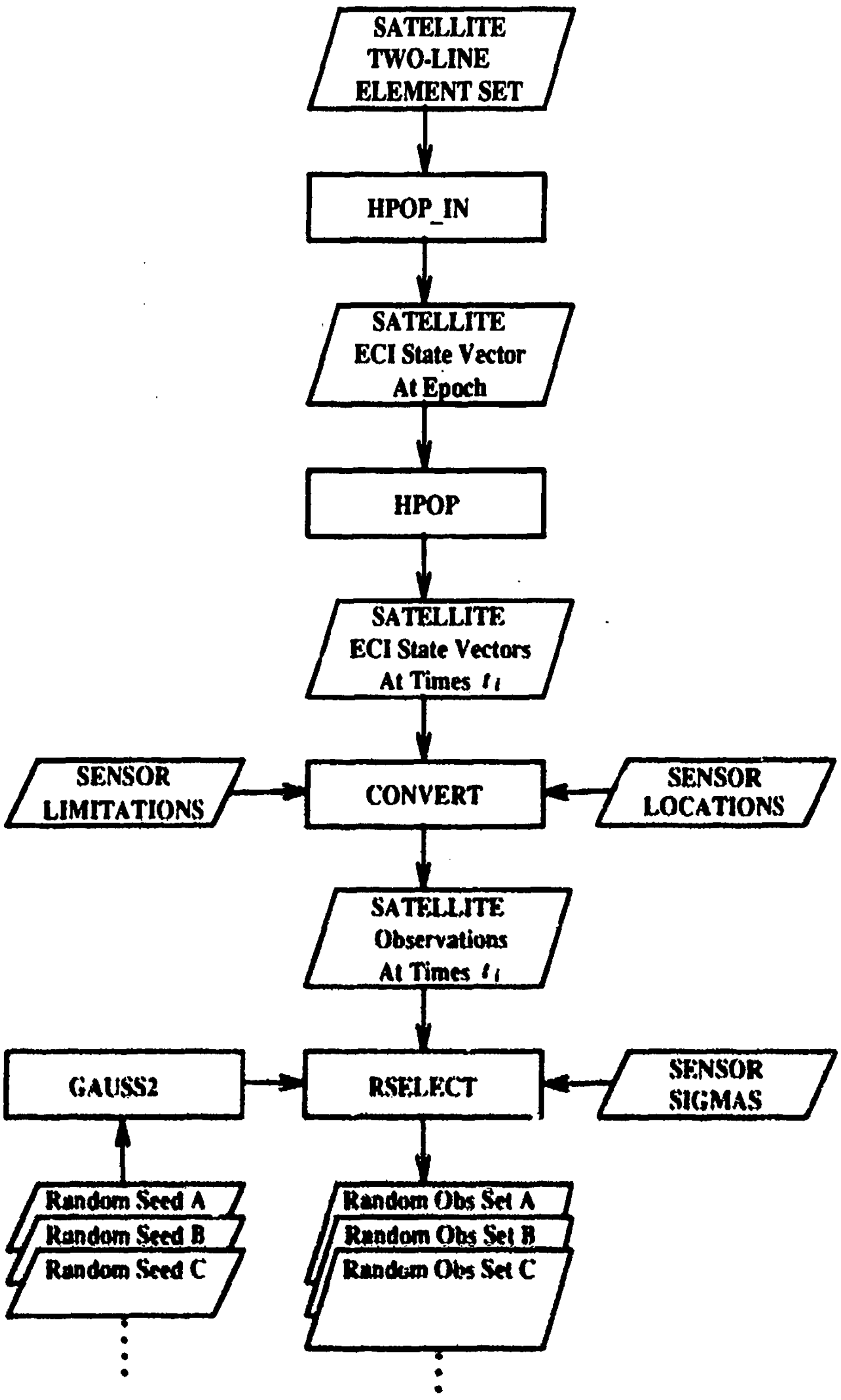


Figure 4.1. Truth Model Flow.

4.3.2.1 HPOP_IN. The program HPOP_IN accepts as input a *.2LE file containing at least one two line element set of the format seen in Appendix F.6. It utilizes a user-friendly windows-style interface to allow easy selection of the *.2LE input file, the propagation step size, and the time interval for propagation. Its output consists of SV-satnr.els files, where SV- indicates a "state vector" file, and satnr and els indicate respectively, the satellite and element set number used to create the position/velocity state vector. HPOP_IN creates this file, formated as seen in Appendix C.2, for each of the two line element sets in the input *.2LE file. The Turbo Pascal source code listing for this program is in Appendix C.1

4.3.2.2 HPOP. HPOP accepts as input the SV-satnr.els state vector file, produced by HPOP_IN. It is executed using the following execution command format:

hpop < SV-satnr.els > PV-satnr.els

where PV- indicates the HPOP created output file containing the propagated, time-tagged, ECI position and velocity vectors for the input satellite data. Reference Appendixes C.2 and C.3 for the format and an example of HPOP input files. A sample of an HPOP output file is in Appendix C.4.

4.3.2.3 CONVERT. This program accepts as input the HPOP created PV-satnr.els file of the format seen in Appendix Section C.4. It reads data from a *.OBS file containing SSN sensor locations and a *.LIM file containing SSN sensor observation limits. It uses a user friendly windows style interface allowing easy selection of the *.OBS, and *.LIM data files, and the PV-satnr.els input file. Its output consists of an OB-satnr.els file, formated the same as the RSELECT output sample shown in Appendix Section C.8, containing the observing sensor, observation time, observations (azimuth or right ascension, elevation or declination, range, and range rate), and the truth ECI position (X, Y, and Z). The Turbo Pascal source code for this program is located in Appendix C.5.

4.3.2.4 RSELECT. This program accepts as input a CONVERT created OB-satnr.elr file. It uses a user friendly windows-style interface allowing easy selection of this

input file. The maximum observation rate and number of random sets of observations desired are entered as command line parameters in the following format:

RSELECT nn mm

where nn is the maximum observation rate (even number of observations/day) desired and mm is the number of sets of random observations (1 to 26) desired. It's output consists of mm different RO-satnr.nnX files for each of $nn/2$ different observation rates. In this file name nn indicates the observation rate and X is an alphabetic index (A, B, C, ..., Z) corresponding to different randomly-selected sets of observations. For example, if a maximum observation rate of 10 per day ($nn = 10$) and 10 random sets ($mm = 10$) is specified, RSELECT will create 10 random observation files (A through J) each for observation rates 2, 4, 6, 8, and 10. The output is formatted the same as shown in Appendix C.8. The Turbo Pascal source code listing of this program is in Appendix C.6.

4.3.3 Model Verification. The truth model was verified as two components. The first component was the HPOP program itself. The second component consisted of the three Turbo Pascal interface programs, HPOP IN, CONVERT, and RSELECT.

The HPOP program is an off-the-shelf, commercial, orbital propagation program. The accuracy of the program was advertised to be on the order of 12 meters or better (15:1). However, we had no means available to verify this level of accuracy. To test the program, ECI position and velocity vectors from HPOP were compared to the output of *SGP4SDP4* for various test satellites. The results of this comparison indicated the level of accuracy would be sufficient for the purpose of our research. This program may be replaced by some other special perturbations code if it cannot be validated as suitable for this purpose.

The HPOP_IN, RSELECT, and CONVERT programs were verified by Dr. Kelso prior to our use. However, to ensure proper performance of the programs, we stepped through each procedure and watched execution using the debug capability of Turbo Pascal.

4.4 Differential Corrector

The purpose of this model is to differentially correct the orbital elements for a satellite using observations created from the truth model, and output a time-tagged vector magnitude of the error between the "truth" position and the calculated, or predicted, position (VMAGT) for each observation. The specific elements corrected will be from the NORAD two-line element set: bstar (B^*), inclination (i_0), right ascension of the ascending node (Ω_0), eccentricity (e_0), argument of perigee (ω_0), mean anomaly (M_0), and mean motion (n_0). Because the results from this model will be compared to actual data from USSPACECOM, the model will attempt to model the composition and characteristics of the SSN as realistically as possible. Additionally, it will use the same SGP4/SDP4 orbital models used by USSPACECOM.

4.4.1 Model Development. The differential corrector was coded using Turbo Pascal, Version 6.0, primarily due to the existence of the SGP Libraries' validated Turbo Pascal code for the USSPACECOM/NORAD orbital models, SGP4 and SDP4 (presented in Section 4.2). This model consists of two primary elements: the actual differential corrector (DC) algorithm and the orbital element/covariance-propagation algorithm. Additionally, various secondary elements providing for the input and output of required data and variable initialization and manipulation are included in the model.

Upon initiation, the model first reads the data files containing the SSN sensors location data (latitude, longitude, and altitude) and weighting data (observation sigmas) as presented in Tables 2.2 and 2.6. The estimated state is initialized to the same orbital elements used to initialize the truth model. The estimated state covariance matrix is also initialized.

It was necessary to "guess" a value of the initial state covariance estimate which would give some confidence in the initial state estimate[3] and keep the "start up" time to a reasonable level. However, we did not want to give too much confidence in the state estimate and prevent the model from correcting the elements. After examination of the covariances of various test satellites, a value of 1.0×10^{-7} was chosen. For simplicity, a

[3] The same elements used to genetate the truth observations were input as the initial state estimate

diagonal matrix was used and coded to allow easy changes if results showed that the initial "guess" was severely affecting the "start-up" time.

Once the initializations are complete, the first batch of observations is read into the program. The batch size (number of observations used in one correction) is equal to the product of the specified observation rate in observations per day (OPD), and the specified correction interval, or length of update interval (LUPI), in days. For example, given an OPD of 4 (observations-per-day) at a LUPI of 2 (days), the batch size would be 8 observations. As the observations are input, a check is performed to determine the observation type (reference Table 2.15) based on the sensor reporting the observation.

The predicted VMAGT for each observation is then calculated. The observation time and the current state estimate is passed to SGP4/SDP4, which calculates a predicted ECI position (r_{x_p},r_{y_p},r_{z_p}). VMAGT is calculated as the simple vector magnitude of the difference between the this ECI position and the truth ECI position (r_{x_t},r_{y_t},r_{z_t}) as shown in the following equation.

$$\mathrm{VMAGT} = \sqrt{(r_{x_p} - r_{x_t})^2 + (r_{y_p} - r_{y_t})^2 + (r_{z_p} - r_{z_t})^2} \tag{4.1}$$

The estimated state and covariance are then propagated to the new epoch time. The time of last ascending node passage prior to the last observation processed was chosen as the new epoch time because this is how USSPACECOM propagates the majority of satellite epochs (24:259). Because of simplifications used in the development of the propagation routine, described in detail in Section 4.4.1.2 below, we expected the model to induce some error in the propagated state and covariance. Because of this known source of error, the estimated state and covariance were propagated prior to differential correction. This allowed the DC to correct the errors induced by the propagation routine and provide the "best" possible estimate for the calculation of VMAGT.

A batch of observation data and the propagated state and covariance are passed to a differential correction routine, described in detail in Section 4.4.1.1 below. Upon convergence, the DC returns a corrected state and covariance, which are then declared the new estimates. Another set of VMAGTs is calculated for the current observation batch

based on the newly corrected state. While this data is not used in the analysis, it is useful as a check to ensure the DC is providing "good" corrections. The flow of the dynamics model is shown in Figure 4.2.

4.4.1.1 Differential Correction Algorithm. As explained in Section 2.9 above, USSPACECOM uses a combination of sequential and batch correction methods to correct satellite orbital elements. In this research, we simplified the correction methodology to use only a Bayes, sequential estimation, correction method as presented in Section 3.2.

The correction routine begins when passed the current estimated state, $\hat{\mathbf{x}}$, and covariance along with a batch of observations, $\hat{\mathbf{O}}_i$. These observations are indexed by time and the sensor producing the observations. A reference state, $\mathbf{x}^*$, is set equal to the current estimated state, $\hat{\mathbf{x}}$. For each observation time t_i, a predicted set of observations, $\mathbf{O}_i$, is calculated using SGP4/SDP4. SGP4/SDP4 calculates the predicted observations as a function of t_i and the current reference state, $\mathbf{x}^*$.

$$\mathbf{O}_i = \mathbf{O}_i(\mathbf{x}^*(t_0), t_i) \tag{4.2}$$

The residual vector, $\mathbf{r}_i$, is then calculated as shown in Equation 4.3.

$$\mathbf{r}_i = \hat{\mathbf{O}}_i - \mathbf{O}_i \tag{4.3}$$

An observation matrix T_i is then calculated for each t_i. Because the sensors of the SSN report different types of observations (reference Section 2.8), T_i could take on different dimensions for each of the different observation types reported. The SSN sensors we modeled only report three observations types: Three, Two, and Five (reference Section 2.8).

For computational simplicity, T_i was given a constant dimension (4×7) based on Observation Type Three, which reports largest set of data (azimuth, elevation, range, and range rate). The observation matrix for Type Three observations is shown below in

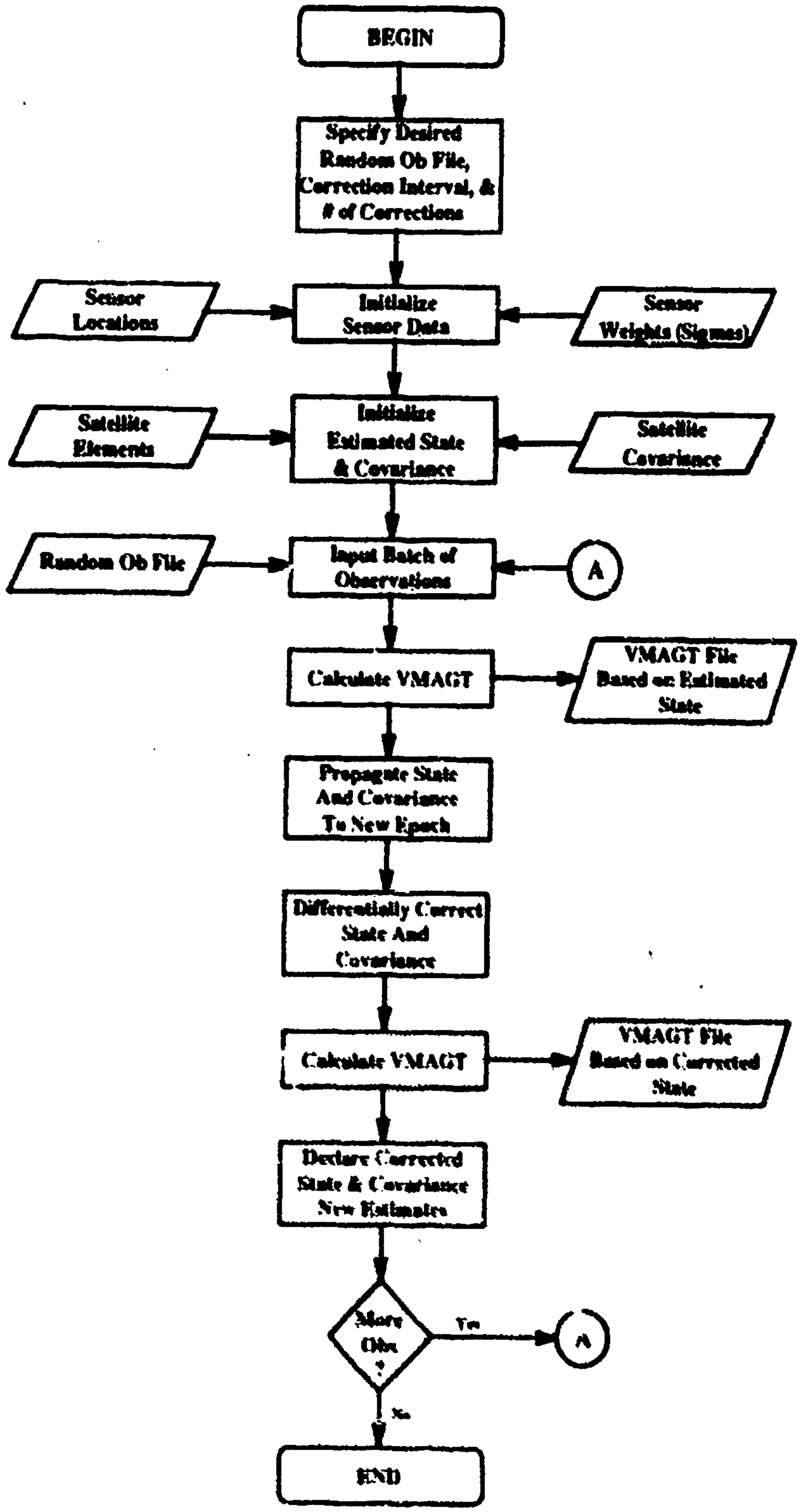


Figure 4.2. Differential Corrector Model Flow.

Equation 4.4.

$$T_i = \begin{bmatrix} \frac{\partial Az(t_i)}{\partial B_0^*} & \frac{\partial Az(t_i)}{\partial i_0} & \frac{\partial Az(t_i)}{\partial \Omega_0} & \frac{\partial Az(t_i)}{\partial e_0} & \frac{\partial Az(t_i)}{\partial \omega_0} & \frac{\partial Az(t_i)}{\partial M_0} & \frac{\partial Az(t_i)}{\partial n_0} \\ \frac{\partial El(t_i)}{\partial B_0^*} & \frac{\partial El(t_i)}{\partial i_0} & \frac{\partial El(t_i)}{\partial \Omega_0} & \frac{\partial El(t_i)}{\partial e_0} & \frac{\partial El(t_i)}{\partial \omega_0} & \frac{\partial El(t_i)}{\partial M_0} & \frac{\partial El(t_i)}{\partial n_0} \\ \frac{\partial \rho(t_i)}{\partial B_0^*} & \frac{\partial \rho(t_i)}{\partial i_0} & \frac{\partial \rho(t_i)}{\partial \Omega_0} & \frac{\partial \rho(t_i)}{\partial e_0} & \frac{\partial \rho(t_i)}{\partial \omega_0} & \frac{\partial \rho(t_i)}{\partial M_0} & \frac{\partial \rho(t_i)}{\partial n_0} \\ \frac{\partial \dot{\rho}(t_i)}{\partial B_0^*} & \frac{\partial \dot{\rho}(t_i)}{\partial i_0} & \frac{\partial \dot{\rho}(t_i)}{\partial \Omega_0} & \frac{\partial \dot{\rho}(t_i)}{\partial e_0} & \frac{\partial \dot{\rho}(t_i)}{\partial \omega_0} & \frac{\partial \dot{\rho}(t_i)}{\partial M_0} & \frac{\partial \dot{\rho}(t_i)}{\partial n_0} \end{bmatrix} \tag{4.4}$$

The observation matrix for Type Two observations (azimuth, elevation, and range) will contain a row of zeros as shown below in Equation 4.5.

$$T_i = \begin{bmatrix} \frac{\partial Az(t_i)}{\partial B_0^*} & \frac{\partial Az(t_i)}{\partial i_0} & \frac{\partial Az(t_i)}{\partial \Omega_0} & \frac{\partial Az(t_i)}{\partial e_0} & \frac{\partial Az(t_i)}{\partial \omega_0} & \frac{\partial Az(t_i)}{\partial M_0} & \frac{\partial Az(t_i)}{\partial n_0} \\ \frac{\partial El(t_i)}{\partial B_0^*} & \frac{\partial El(t_i)}{\partial i_0} & \frac{\partial El(t_i)}{\partial \Omega_0} & \frac{\partial El(t_i)}{\partial e_0} & \frac{\partial El(t_i)}{\partial \omega_0} & \frac{\partial El(t_i)}{\partial M_0} & \frac{\partial El(t_i)}{\partial n_0} \\ \frac{\partial \rho(t_i)}{\partial B_0^*} & \frac{\partial \rho(t_i)}{\partial i_0} & \frac{\partial \rho(t_i)}{\partial \Omega_0} & \frac{\partial \rho(t_i)}{\partial e_0} & \frac{\partial \rho(t_i)}{\partial \omega_0} & \frac{\partial \rho(t_i)}{\partial M_0} & \frac{\partial \rho(t_i)}{\partial n_0} \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix} \tag{4.5}$$

The observation matrix for type five observations (right ascension and declination) will contain a two rows of zeros as shown below in Equation 4.6.

$$T_i = \begin{bmatrix} \frac{\partial \alpha(t_i)}{\partial B_0^*} & \frac{\partial \alpha(t_i)}{\partial i_0} & \frac{\partial \alpha(t_i)}{\partial \Omega_0} & \frac{\partial \alpha(t_i)}{\partial e_0} & \frac{\partial \alpha(t_i)}{\partial \omega_0} & \frac{\partial \alpha(t_i)}{\partial M_0} & \frac{\partial \alpha(t_i)}{\partial n_0} \\ \frac{\partial \delta(t_i)}{\partial B_0^*} & \frac{\partial \delta(t_i)}{\partial i_0} & \frac{\partial \delta(t_i)}{\partial \Omega_0} & \frac{\partial \delta(t_i)}{\partial e_0} & \frac{\partial \delta(t_i)}{\partial \omega_0} & \frac{\partial \delta(t_i)}{\partial M_0} & \frac{\partial \delta(t_i)}{\partial n_0} \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix} \tag{4.6}$$

In this model, the elements of the observation matrix were calculated numerically using Equation 3.88, restated below, as presented in Section 3.3.

$$\frac{\partial O_j(t_i)}{\partial x_k(t_o)} \cong \frac{O_j(\mathbf{x}(t_o) + \Delta x_k(t_o), t_i) - O_j(\mathbf{x}(t_o), t_i)}{\Delta x_k(t_o)} \tag{4.7}$$

The incrementally small $\Delta \mathbf{x}_k$ values used were experimentally determined by comparing the significant figures in the SGP4/SDP4 calculated observations shown below.

$$O_j(\mathbf{x}(t_0), t_i) \tag{4.8}$$

and

$$O_j(\mathbf{x}(t_0) + \Delta x_k(t_0), t_i) \tag{4.9}$$

The matrix sums $T_i^T Q_i^{-1} T_i$ and $T_i^T Q_i^{-1} \mathbf{r}$, are then calculated for each t_i. The sensor covariance matrix, Q_i, is dependent on the sensor reporting the observations. As with the T_i matrix, it will be a different dimension based on the observation type of the sensors, but for the purpose of this model is held at a constant dimension (4×4). Equations 4.10, 4.11, and 4.12 show the weighting matrix for Observation Type Three, Two, and Five, respectively.

$$Q_i = \begin{bmatrix} \sigma^2_{Az\,i} & 0.0 & 0.0 & 0.0 \\ 0.0 & \sigma^2_{El\,i} & 0.0 & 0.0 \\ 0.0 & 0.0 & \sigma^2_{r\,i} & 0.0 \\ 0.0 & 0.0 & 0.0 & \sigma^2_{\dot{r}\,i} \end{bmatrix} \tag{4.10}$$

$$Q_i = \begin{bmatrix} \sigma^2_{A\,i} & 0.0 & 0.0 & 0.0 \\ 0.0 & \sigma^2_{El\,i} & 0.0 & 0.0 \\ 0.0 & 0.0 & \sigma^2_{r\,i} & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix} \tag{4.11}$$

$$Q_i = \begin{bmatrix} \sigma^2_{\alpha\,i} & 0.0 & 0.0 & 0.0 \\ 0.0 & \sigma^2_{\delta\,i} & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix} \tag{4.12}$$

As the matrix sums $T_i^T Q_i^{-1} T_i$ and $T_i^T Q_i^{-1} \mathbf{r}_i$ are calculated for each t_i, the running sums

$$\sum T_i^T Q_i^{-1} T_i \tag{4.13}$$

and

$$\sum T_i^T Q_i^{-1} \mathbf{r}_i \tag{4.14}$$

are maintained.

After all observations have been processed. The new state covariance, P_{new}^{-1}, is calculated as shown in Equation 4.15.

$$P_{new}^{-1} = P_{old}^{-1} + \Sigma T_i^T Q_i^{-1} T_i \tag{4.15}$$

where P_{old}^{-1} is the old, or *a priori*, inverse state covariance.

The state correction estimate is then calculated as shown in Equation 4.16

$$\delta \mathbf{x}(t_0) = P_{new} \left(P_{old}^{-1} (\bar{\mathbf{x}}_{old} - \mathbf{x}^*) + \Sigma T_i^T Q_i^{-1} \mathbf{r}_i \right) \tag{4.16}$$

with the corrected reference state then being calculated as shown in Equation 4.17.

$$\mathbf{x}_{k+1}^*(t_0) = \mathbf{x}_k^*(t_0) + \delta \mathbf{x}(t_0) \tag{4.17}$$

To prevent the DC from attempting to correct certain elements to a value outside their allowable range, some checks and limits routines were implemented. The three elements of concern were B^*, e_0, and n_0. First, a maximum absolute value of 1.0 was set for B^*. If this limit was exceeded, the corrected B^* value was reset to the uncorrected, or previous reference value. This value was chosen as a reasonable value based on the statistical analysis of the current population. Additionally, if this B^* limit is exceeded, B^* is removed from the RMS calculation for convergence.

The maximum limits for e and n were set based on Equations 4.18 and 4.19, which express the maximum value of one element as a function of the other.

$$e_{max}(n_0) = 1 - R_{\oplus}(n_0^2/\mu)^{1/3} \tag{4.18}$$

$$n_{max}(e_0) = (1 - e_0)^{3/2}\sqrt{\mu/R_{\oplus}^3} \tag{4.19}$$

A value of 0.0 was set as the lower limit for each of these elements. A check was first performed on e_0. If outside these limits, the corrected e_0 was reset to the previous reference value. Similarly, if n_0 was outside these limits, the corrected value was reset to the previous reference value.

The correction is then checked for convergence. We used the weighted root mean square (RMS) of the calculated state corrections as shown in Equation 4.20.

$$\text{RMS} = \sqrt{\frac{\delta\bar{\mathbf{x}}^T(t_0)\,P^{-1}(+)\,\delta\bar{\mathbf{x}}(t_0)}{7}} \tag{4.20}$$

If the weighted RMS value is less than 0.01, the process is considered converged. The corrected reference state, $\mathbf{x}_{k+1}^*$, and corrected covariance, $P(+)$, are passed out of the differential correction routine and then declared the new estimates. If convergence is not declared, the differential process begins again using $\mathbf{x}_{k+1}^*$ as the new reference state.

4.4.1.2 Propagation Routine. Prior to performing differential correction, the epoch of the current estimated element set is moved to the time of the last ascending node crossing prior to the last observation processed. To do this, an analytic method was decided upon for computational efficiency and because this is how the majority of satellite epochs are propagated (24:259). Once propagated to the new time, the state covariance matrix is propagated to provide an estimate of how good the new state vector is.

The elements used by SGP are mean Keplerian orbital elements. The mean elements, in a sense, average out the perturbations over an orbit and, thus, are only accurate for longer-term predictions. To avoid any cumulative effects of propagating the orbit to any place in its path, the ascending node crossing is used to provide a consistent point to average

the orbit. This also reflects what is done by the SSC's current differential correction techniques as can be shown in Figures 5.8 thru 5.11. As can be seen, the sum of the true anomaly (ν) and the argument of perigee (ω) is fairly close to 0 or 2π only when the inclination and the eccentricity are close to 0 does the sum of the two appear to be fairly random. The divergence at higher eccentricities is due to convergence problems of extracting the true anomaly from the mean anomaly.

To propagate the mean Keplerian orbital elements, we decided to initially perform element propagation for J_2 perturbations only. Also, the air drag and resonance effects are not considered. Additionally, we made no attempt to incorporate the $\dot{n}$ and $\ddot{n}$ terms into the propagation. Section 3.1 explains the derivation of the propagated elements and the state transition matrix.

Since the longest propagation time is going to be on the order of ten days, the exclusion of the resonance terms is expected to have only a small impact. For satellites in a very low orbit, the assumption that there is no air drag is not expected to be a good one. To account for the dynamics model errors, a de-weighting or forgetting factor is used in deciding how good the estimate is The forgetting factor is actually a 7×7 matrix that is a function of propagation time. Additionally the forgetting factor could be modified to account for the air drag and resonance shortcomings of the propagator. However, we decided against implementing this due to our time restrictions. We made another simplification in our model to assume the forgetting matrix was a diagonal matrix with all the elements being identical, this effectively reduces our forgetting matrix into a forgetting scalar. Reference (24:202) provides the baseline forgetting scalar, used by USSPACECOM, that we will use, providing it does not over emphasize or deemphasize the estimate. Over emphasis will manifest itself as an estimate which refuses to change thus the first and last pass residuals grow. Over deemphasis will be estimates which fluctuate wildly. The last pass residuals will be low but first pass residuals will be high.

4.4.2 Model Execution. The primary program implementing this model is called DIFC. Six additional Turbo Pascal units provide routines necessary to support the main

program DIFC. Source code listings for these units, described below, can be found in Appendix D.

- *DC_INIT.* Unit containing constants, variables, and declarations needed to initialize the DC and interface the DC with the other routines.
- *DC_CALC.* Unit containing routines for performing the differential correction calculations and variable manipulations required by DC.
- *PROP.* Unit containing routines for propagating the elements and their covariance.
- *LOWB.* Unit containing routines for reading SSN sensor locations and weights data.
- *DC_OUT.* Unit containing routines to output program results in standard formats.

DIFC accepts as input a 2LEsatnr.DAT file containing the initial estimated state (orbital elements), a COVsatnr.DAT file containing the initial estimated state covariance, and a RO-satnr.nnX file, specified in a command line parameter, containing random observations. Recall that this file contains random observation set X, at observation rate nn, for satellite satnr. DIFC automatically reads the SENSORS.OBS and SENSORS.COV data files containing the SSN sensor location and weighting data. The correction interval and desired number of corrections are specified as command line parameters in the following format:

DIFC RO-satnr.nnX aa bb

where aa is the desired correction interval in days and bb is the desired number of corrections to be performed. Be aware that the product aa×bb must be less than the total number of days of observation data in the RO-satnr.nnX file.

DIFC outputs three different files:

- File Fsatnrnn.aaX containing observation time (expressed in days since epoch of the original estimate elset) and the VMAGT prior to differential correction.
- File Lsatnrnn.aaX containing observation time (expressed in days since epoch of the original estimate elset) and the VMAGT after differential correction.

- File Rsatnrnn.aaX containing error flags, initial and corrected elements and covariances, and any other information not directly required for the statistical analysis, but of possible use in analyzing adverse or interesting trends appearing in the data.

where satnr indicates the satellite number, nn indicates the observation rate in obs/day, aa indicates the correction interval in days, and X is an alphabetic index corresponding to different random observation sets.

4.4.3 Model Verification. Verification of program algorithms and data flow were performed by checking sample case calculations with either independent programs or hand calculation. The following verification procedures were used:

- Matrix and Vector Operations. Intensive verification of all matrix and vector operations used within our differential corrector model was performed using the MatLab[4] numeric computation software package, QuatroPro[5] spreadsheet calculation software, and in some cases hand calculations. Test routines were placed in the program to print the values of matrix and vector elements prior to, and following, all operations. These same input values were used to perform the same matrix or vector operations using one or both of the software packages discussed above. This procedure was performed several times, for different test cases, throughout the development of the model.

- Numerical Calculation of Observation Matrix Elements. Numerical accuracy of the variables used to calculate the elements of the observation matrix were verified manually. Test routines were placed in the programs to print the values of the variables used to calculate the elements of the T_i matrix ($O_j(\mathbf{x}(t_0), t_i)$, $O_j(\mathbf{x}(t_0) + \Delta x_k(t_0), t_i)$, and $\Delta x_k(t_0)$). The number of significant digits present in the difference calculations were examined.

- Dynamics Model Flow. The proper data flow of information in the program, as well as the input and output of data by the program was verified by examining

[4]MatLab is a trademark of The MathWorks, Inc.
[5]QuatroPro is a trademark of Borland International.

various test print routines and performing a line by line program trace monitoring various variables. This procedure was performed several times, for different test cases, throughout the development of the model.

4.5 *Statistical Analysis.*

The overall purpose of this research is to develop and demonstrate a model from which the effects of observation rate and update intervals on the accuracy of an orbital element set can be determined. Therefore, the purpose of this analysis is to see if the steady-state VMAGT value can be determined for the various observation rate update interval combinations tested, and if the value can be determined, show how observation rate and update interval affect it. The following sections will review how the values of observation rate and correction interval tested were determined, how the number of Monte Carlo runs was determined, and discuss the variance reduction and data analysis techniques used.

4.5.1 Observation Rate/Update Interval Selection. The differential corrector model will be run using various observation rates and correction interval combinations for various sample satellites. A review of the average observation rates recommended in the System Capability Study (reference Table 2.9) and used by the USSPACECOM tasking program, as described in Section 2.7, show required average observation rates of between one and five observations-per-day. A review of the typical observation tasking for routine satellites, as shown in Table 2.12, shows current observation tasking on routine satellites being between approximately four and nine observations-per-day. Based on these values, we chose to examine observation rates of two, four, six, and eight observations-per-day.

The SSC's automatic element set update process flags a routine satellite for a differential correction every 24 hours (reference Section 2.9.2.1). Based on this data, we decided to examine update intervals of two, four, six, and eight days.

4.5.2 Monte Carlo Analysis. One purpose of simulation is to thoroughly examine an alternative by generating values for the random variable at frequencies indicated by the

distribution of the random variable (20:494). Of interest to this research is the random variable representing the vector magnitude difference between the computed position and actual position of a satellite. This quantity is referred to as VMAGT. The measurement of this variable may be treated as a random variable because of the inaccuracies involved with the sensor devices. These inaccuracies are expressed as average error (bias) and standard deviation (sigma) and are sensor dependent. The result is a correctable measured position with a definable variation.

In order to gain some insight into the ability of our differential corrector to correct orbital elements, Monte Carlo simulations will run for the various combinations of correction interval (LUPI) and number of batches used for correction. Observations will be calculated for a period of 60 days. To test the ability of the differential corrector, a factorial experiment will be created. With correction intervals of every two, four, six, and eight days, this gives 30, 15, 10, or 7 batches, respectively. For each of these combinations, the differential corrector will be run based on an observation rate of two, four, six, and eight observations per day.

Of major concern to the simulation is the number of trials for each factor combination required to gather the necessary data. The very precision and reliability of the estimate is determined by the number of simulation trials. To estimate the number of trials, we utilized a method suggested by Lapin (20:508-509).

First, we *guessed* a value for the sample deviation. Based on the assumption of a uniform distribution, Lapin suggests the following rule of thumb for making such a guess:

$$\sigma_g = \frac{\text{Largest Value} - \text{Smallest Value}}{6} \qquad (4.21)$$

Since the SSC flags observations for element set update if the position error is 14 kilometers or greater, we set our largest value to 14 km. The smallest possible value is 0 km and indicates a perfect prediction of the satellite position with respect to the measured position. Based on Equation 4.21, $\sigma_g = 2.333$ kilometers.

To determine n, the number of trials, the levels of precision (d) and reliability (z) must be established. Lapin defines precision as the maximum deviation from the true value

the experimenter is willing to accept. z is the normal deviate for the required reliability. Lapin provides, without proof, the following formula for determining the required number of trials:

$$n = \frac{z^2\sigma_g^2}{d^2} \tag{4.22}$$

As an initial trial, we select a value for d, based on 10 percent of the largest value. Therefore, $d = 1.4$ kilometers. At the .90 reliability level, the normal deviate is $z = 1.65$. Based on these choices, Equation 4.22 gives a value

$$n = \frac{(1.65)^2(2.333)^2}{(1.4)^2} = 7.56 \approx 8$$

At the .95 reliability level, $z = 1.96$ and the value for Equation 4.22 becomes $n = 10.67$. For the purpose of our Monte Carlo simulation, we chose to execute 10 runs for each case.

4.5.3 Variance Reduction. Of the two variance-reduction methods discussed in Section 3.4, the only variance-reduction technique we will use is common random numbers. As discussed in Section 3.4, a fundamental requirement for using AVs is that the random variables must be monotonic. After some inspection of our model, we determined the random numbers are not monotonic for the following reasons:

- For the observation selections, it is expected that an observation that fell relatively close in time to another in one replication, would not behave any differently in its antithetic pair replication since both observations would occur at a different time but still close to each other. The maximum benefit would occur if the observations were spread out (supposedly).
- For the observation noise, an error equally in one direction probably would not behave much differently than its antithetic pair observation. A negative large error would have an antithetic pair error of a large positive error both impact the system

negatively. In general, observations closer to the mean behave better than ones more distant.

4.5.4 Data Analysis. The major measure of performance of the differential corrector model for the various satellite/LUPI/OPD combinations tested is the steady-state VMAGT value. To determine this value, the point at which steady-state was reached had to first be declared. This declaration was set at the 24th day unless a visual inspection of the raw data plotted with a four-day moving average showed that steady-state occurred later. From the point at which steady-state is declared, the VMAGT mean and variance for each of the 10 Monte Carlo runs was computed as shown in Equations 4.23 and 4.24

$$\overline{\mathrm{VMAGT}_x} = E(\mathrm{VMAGT}_x) = \frac{\sum_{i=1}^{N} \mathrm{VMAGT}_{x_i}}{N} \tag{4.23}$$

$$Var(\mathrm{VMAGT}_x) = \frac{\sum_{i=1}^{N}(\mathrm{VMAGT}_{x_i} - \overline{\mathrm{VMAGT}_x})^2}{N-1} \tag{4.24}$$

where x indicates the one of the 10 different Monte Carlo runs (A thru J), and N is the total number of VMAGT measurements taken since steady-state was declared for a given runs.

A 99 percent confidence level (99% CL) for each VMAGT ($99CL(\mathrm{VMAGT}_x)$) was then calculated for each of the 10 runs as shown in Equation 4.25

$$99CL(\mathrm{VMAGT}_x) = \overline{\mathrm{VMAGT}_x} + 2.32635\left(\sqrt{Var(\mathrm{VMAGT}_x)}\right) \tag{4.25}$$

where 2.32635 represents the one-sided standard normal 99 percent confidence level.

Using variations of Equations 4.23 and 4.24 with the mean, variance, and 99 percent confidence level values calculated from all 10 Monte Carlo runs (x = A through J), the following means and variances were then calculated:

$E(E(\mathrm{VMAGT}_x))$	$Var(E(\mathrm{VMAGT}_x))$
$E(Var(\mathrm{VMAGT}_x))$	$Var(Var(\mathrm{VMAGT}_x))$
$E(99CL(\mathrm{VMAGT}_x))$	$Var(99CL(\mathrm{VMAGT}_x))$

Recall that the $100(1-\alpha)$ confidence interval for any value $Z(n)$ can be calculated as shown below:

$$E(Z(n)) \pm z_{\alpha/2}\left(\sqrt{Var(Z(n))}\right) \tag{4.26}$$

Using Equation 4.26 with $\alpha = 0.05$ ($z_{\alpha/2} = 1.96$) the 95 percent confidence intervals for $E(\text{VMAGT}_x)$, $Var(\text{VMAGT}_x)$, and $99CL(\text{VMAGT}_x)$ are calculated as shown in the three equations below:

$$E(E(\text{VMAGT}_x)) \pm 1.96\left(\sqrt{Var(E(\text{VMAGT}_x))}\right) \tag{4.27}$$

$$E(Var(\text{VMAGT}_x)) \pm 1.96\left(\sqrt{Var(Var(\text{VMAGT}_x))}\right) \tag{4.28}$$

$$E(99CL(\text{VMAGT}_x)) \pm 1.96\left(\sqrt{Var(99CL(\text{VMAGT}_x))}\right) \tag{4.29}$$

The results from Equation 4.25 were also used in the performance of an analysis of variance (ANOVA). For each LUPI/OPD/Monte Carlo run, the 99 percent CL was calculated. Using LUPI and OPD as main effects, a two factor ANOVA was conducted on the 99 percent CL for each satellite. This analysis allowed quantitative determinations concerning the effects of LUPI and OPD on observed VMAGT.

Once all of this statistical data is computed, the data from each sample satellite will be examined with respect to four performance measures:

1. "First pass" VMAGT values
2. "Last-pass" VMAGT values
3. OPD effect on the 99 percent CL.
4. LUPI effect on the 99 percent CL.
5. Statistical differences in OPD, LUPI, and OPD/LUPI interaction effects.

V. ANALYSIS OF EARTH SATELLITE POPULATION

The analysis portion of the earth satellite population was divided into four categories. The first section deals with the composition of the current population.[1] The second section deals with an analysis of the J_2 perturbations. The third section deals with the development of a satellite classification scheme based on orbital elements. The fourth section combines the results of population, J_2, and classification analysis to select representative satellites based on our classification scheme.

5.1 Satellite Composition Analysis

Included in Appendix E are the various graphs which were developed as part of the satellite composition analysis. We have included the more interesting findings in this chapter. The analysis was done on the most current data available. Appendix F contains the code and some brief descriptions as to how the current satellite population data set was determined.

Table 5.1 shows the populations of each of the 32 Gabbard Classes. The most populace classes have altitudes less than 2000 km, except for Class 32, which includes the geosynchronous satellites, and Classes 17 and 26, which, in general, are transfer vehicles. Figure 5.1 shows the density of satellites in Classes 1–16.

It is difficult to make any conclusions about the distributions of satellites in the Gabbard classes. However, the relative numbers indicate most of the satellites fall into eight different classes.[2]

Besides the distributions of satellites in the Gabbard classes, we analyzed the distributions of the two-line elements in the current population. Table 5.2 gives the minimum, mean, median, and maximum for each of the orbital elements in the two-line element sets.

Figures 5.2 through 5.4 show the distributions of eccentricity, inclination, and Kozai mean motion. The distributions of the other elements are in Appendix E.

[1] Current population refers to the first two line element set after 1 March 1990. The 1990 population is assumed to represent the current population of satellites.

[2] 5108 satellites out of 6092 fall into eight classes.

Table 5.1. Satellite Population Within Each Gabbard Class.

Gabbard			Apogee Altitude		Perigee Altitude	
Class	Qty.	Pct.	Min	Max	Min	Max
		%	km	km	km	km
1	189	3.10	0	575	0	575
2	153	2.51	575	1000	0	575
3	1635	26.85	575	1000	575	1000
4	295	4.84	1000	2000	0	575
5	939	15.41	1000	2000	575	1000
6	1378	22.62	1000	2000	1000	2000
7	77	1.26	2000	3000	0	575
8	136	2.23	2000	3000	575	1000
9	85	1.40	2000	3000	1000	2000
10	4	0.07	2000	3000	2000	3000
11	37	0.61	3000	5555	0	575
12	47	0.77	3000	5555	575	2000
13	32	0.53	3000	5555	2000	3000
14	8	0.13	3000	3700	3000	3700
15	30	0.49	3700	5555	3000	3700
16	6	0.10	3700	5555	3700	5555
17	173	2.84	5555	35000	0	575
18	68	1.12	5555	35000	575	2000
19	4	0.07	5555	35000	2000	3000
20	0	0.00	5555	35000	3000	3700
21	0	0.00	5555	35000	3700	5555
22	4	0.07	5555	11110	5555	11110
23	4	0.07	11110	35000	5555	11110
24	79	1.30	11110	35000	11110	35000
25	76	1.25	35000	∞	0	575
26	154	2.53	35000	∞	575	2000
27	58	0.95	35000	∞	2000	3000
28	25	0.41	35000	∞	3000	3700
29	34	0.56	35000	∞	3700	5555
30	3	0.05	35000	∞	5555	11110
31	16	0.26	35000	∞	11110	35000
32	343	5.63	35000	∞	35000	∞
ALL	6092	100.0	0	∞	0	∞

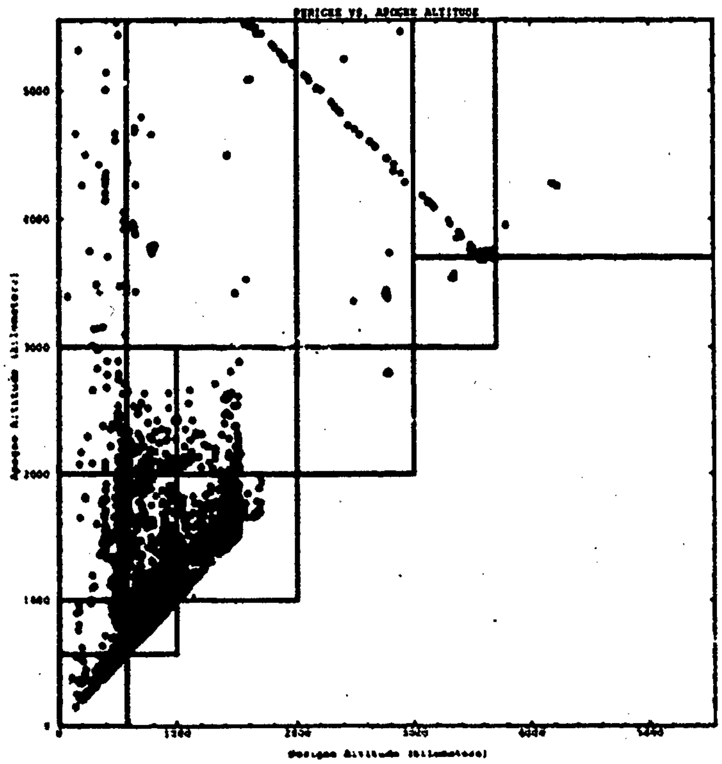


Figure 5.1. Distribution of Satellites in Gabbard Classes 1–16.

Table 5.2. Orbital Element Statistics (Minimum, Mean, Median, Maximum).

Two-Line Element	Units	Minimum	Mean	Median	Maximum
$n_{s,0}$	rev/day	0.096	11.561	13.456	16.478
$\dot{n}_0/2$	rev/day^2	−0.16062	5.20×10^{-4}	8.68×10^{-6}	0.47098
$\ddot{n}_0/6$	rev/day^3	-4.00×10^{-5}	6.37×10^{-5}	0.	0.22599
i_0	deg	0.01	72.26	74.03	144.64
Ω_0	deg	0.00	172.98	167.98	359.97
ω_0	deg	0.00	187.17	194.28	359.86
e_0		1.96×10^{-5}	0.07944	0.00715	0.92278
B_0^*	m^2/kg	−1.608	0.00481	7.36×10^{-4}	0.99831
M_0	deg	0.02	168.98	163.43	359.95

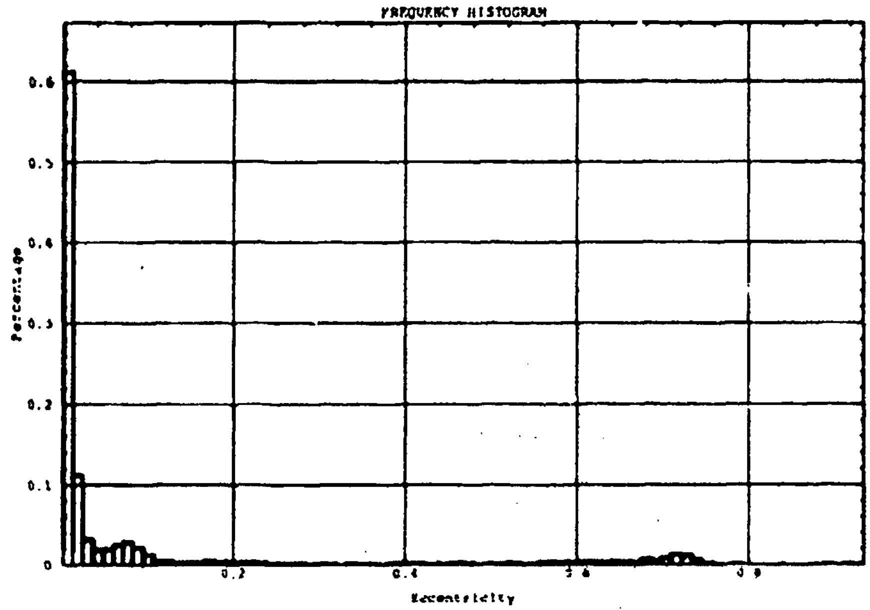


Figure 5.2. Frequency Histogram Eccentricity.

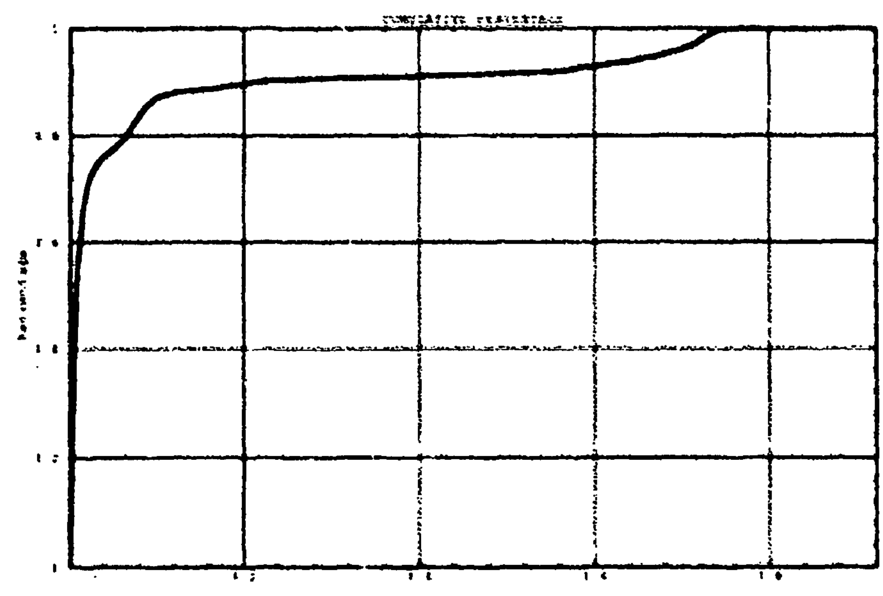


Figure 5.3. Cumulative Distribution Eccentricity.

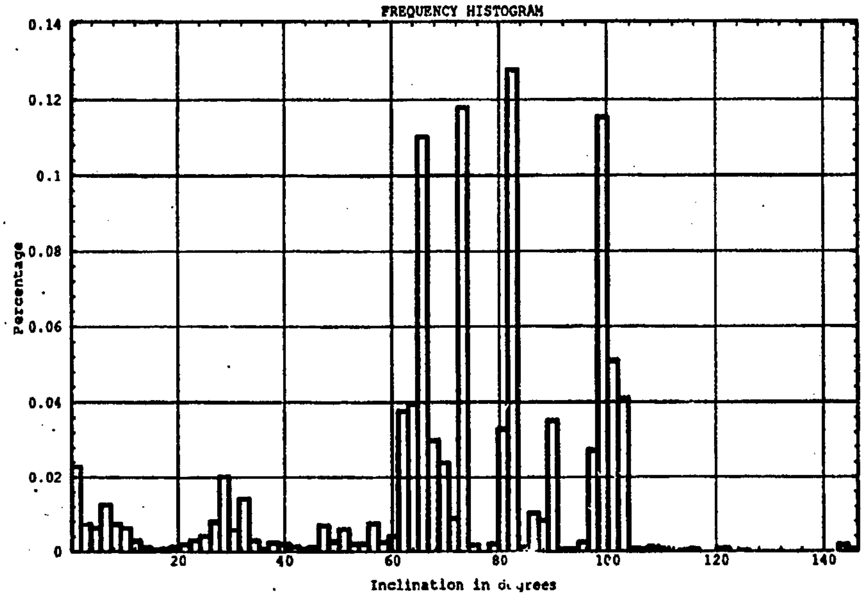


Figure 5.4. Frequency Histogram — Inclination.

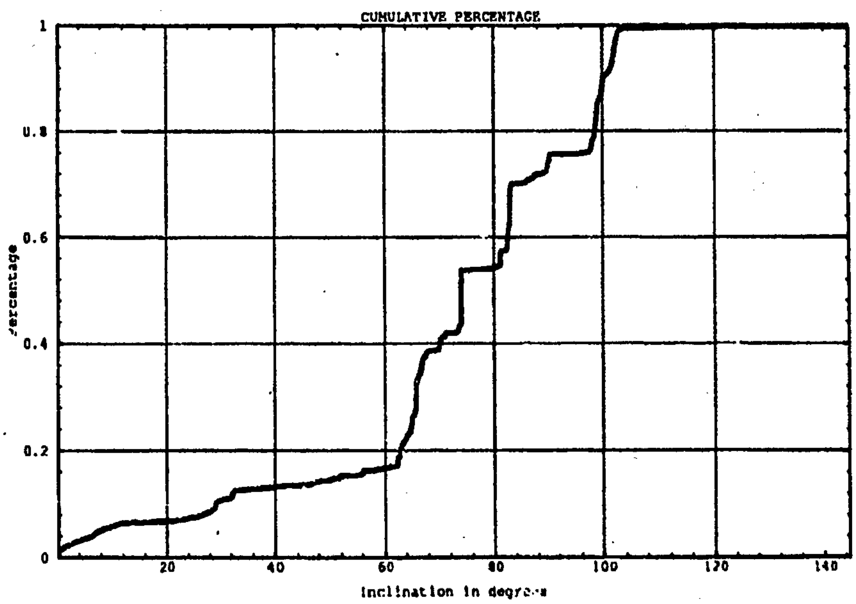


Figure 5.5. Cumulative Distribution — Inclination.

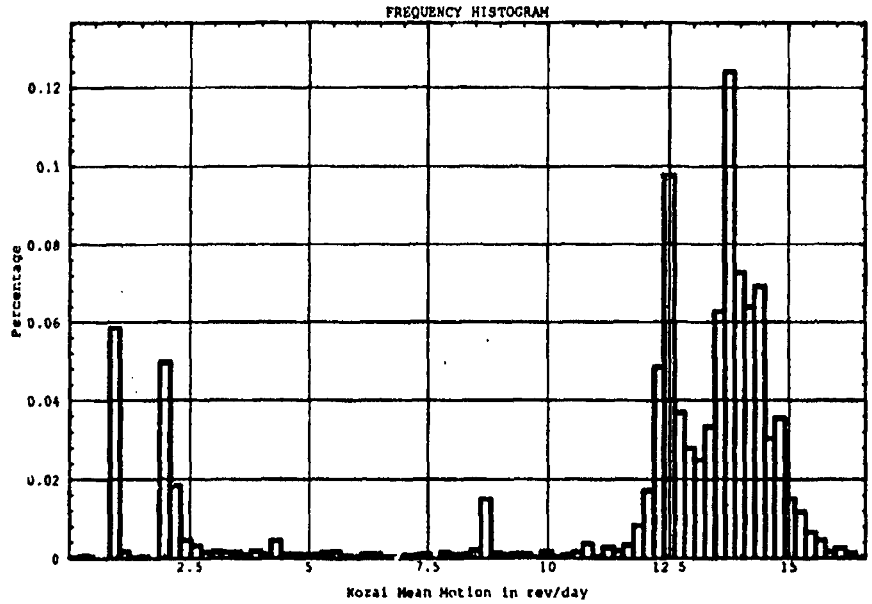


Figure 5.6. Frequency Histogram — Kozai Mean Motion.

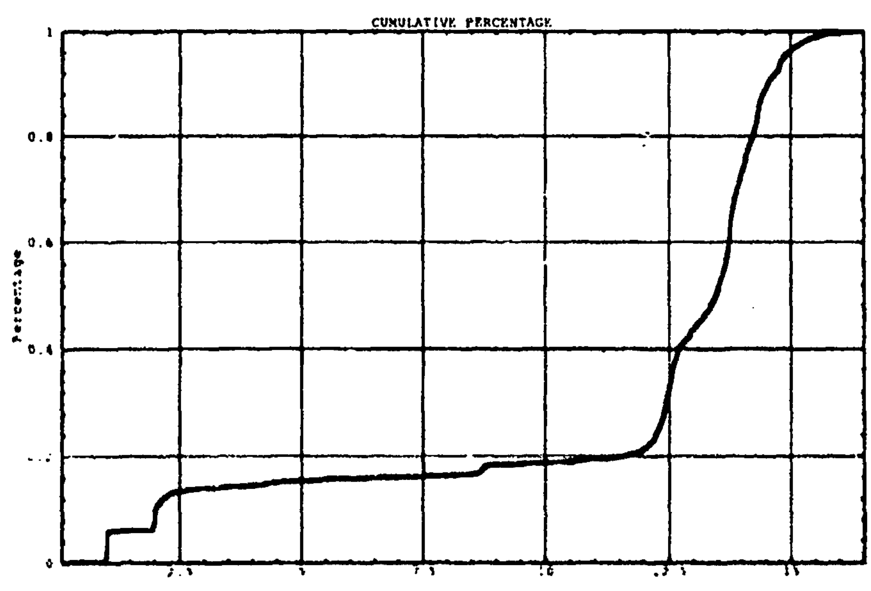


Figure 5.7. Cumulative Distribution Kozai Mean Motion.

After examining the single variable cases we generated some two-variable plots. These plots show if there is a relation between any two orbital elements. For instance, Figure 5.8 shows a correlation between the argument of perigee and mean anomaly.

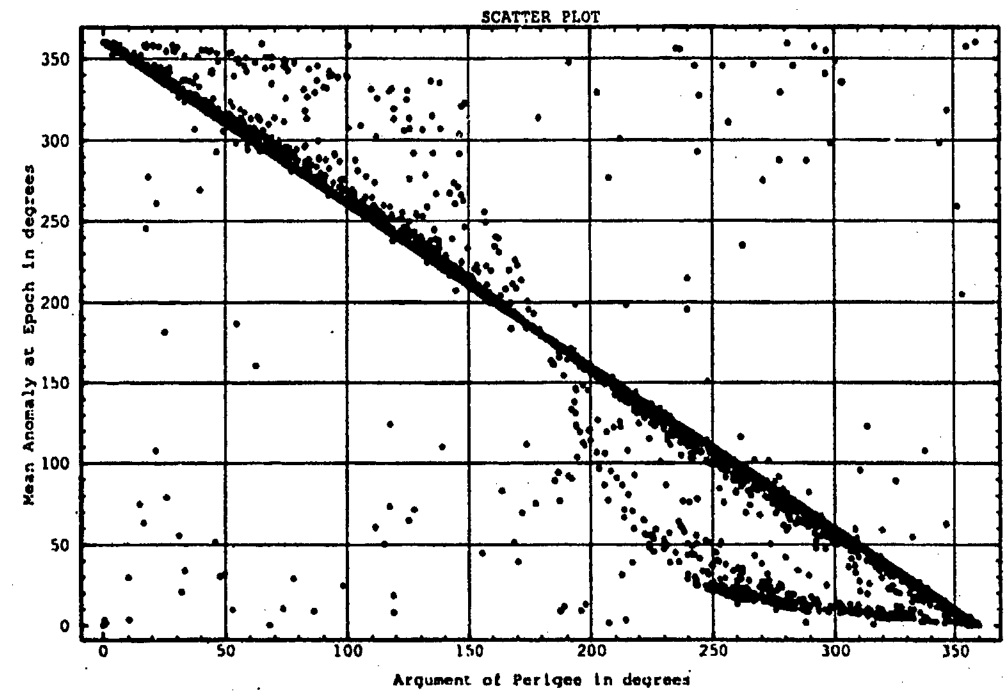


Figure 5.8. Scatter Plot - Argument of Perigee and Mean Anomoly.

Figure 5.9 shows that the correlation is even more pronounced when the mean anomaly at epoch is converted to the true anomaly at epoch.[3]

This correlation indicates the sum of the true anomaly at epoch and the argument of perigee is nearly constant. This finding is consistent with the ICACS procedure of propagating an orbit to its ascending node crossing. The only case where the ascending node crossing is not well defined is when the inclination is near zero. Figure 5.10 shows the relation between the inclination and the angular difference from the node.

As Figure 5.10 shows, most of the large deltas from the node occur at near zero inclination. There is error introduced by performing a power series expansion to change

[3]Some errors are introduced in extracting the true anomaly from the mean anomaly. The errors are greatest when eccentricity is large.

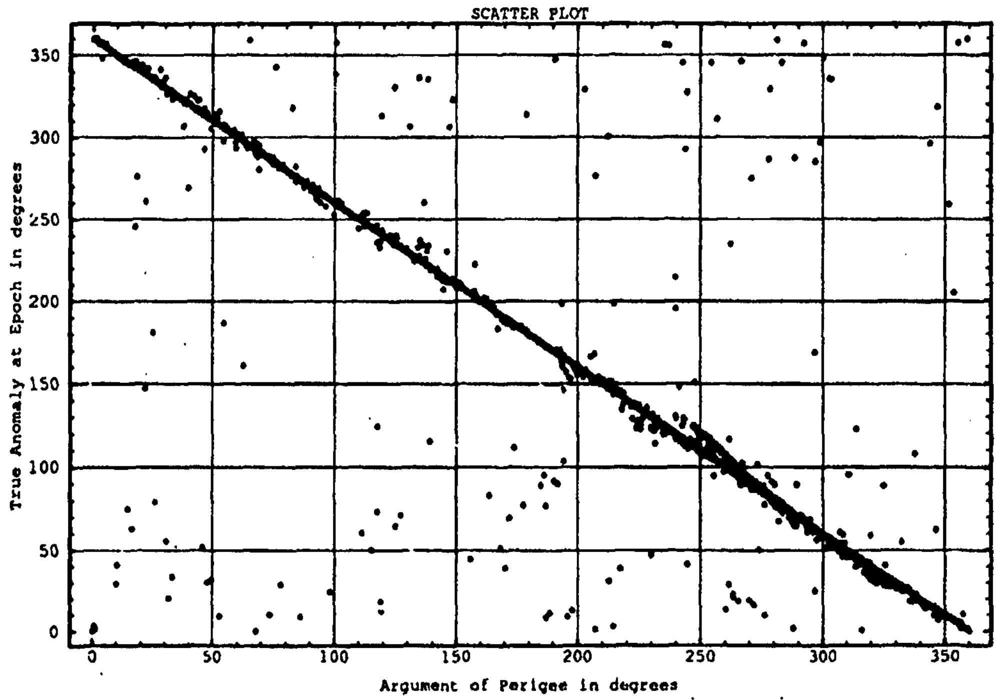


Figure 5.9. Scatter Plot -- Argument of Perigee and True Anomaly.

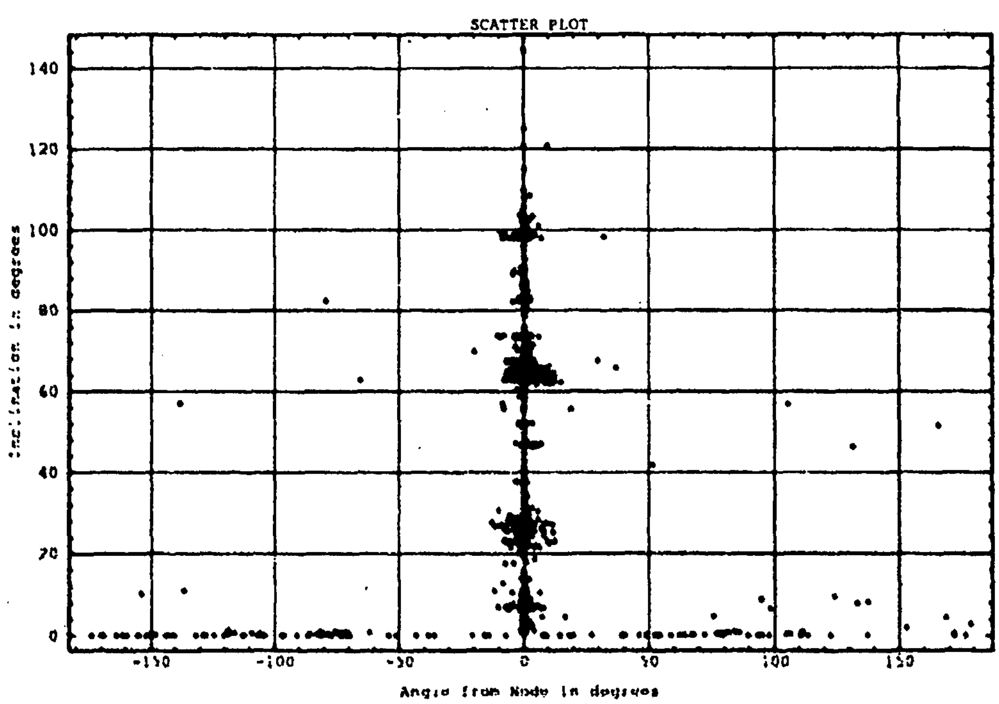


Figure 5.10. Scatter Plot Angle from Node and Inclination.

the mean anomaly into the true anomaly and this is more pronounced at the higher eccentricities. Figure 5.11 provides some idea as to how large the error is in the power series expansion.

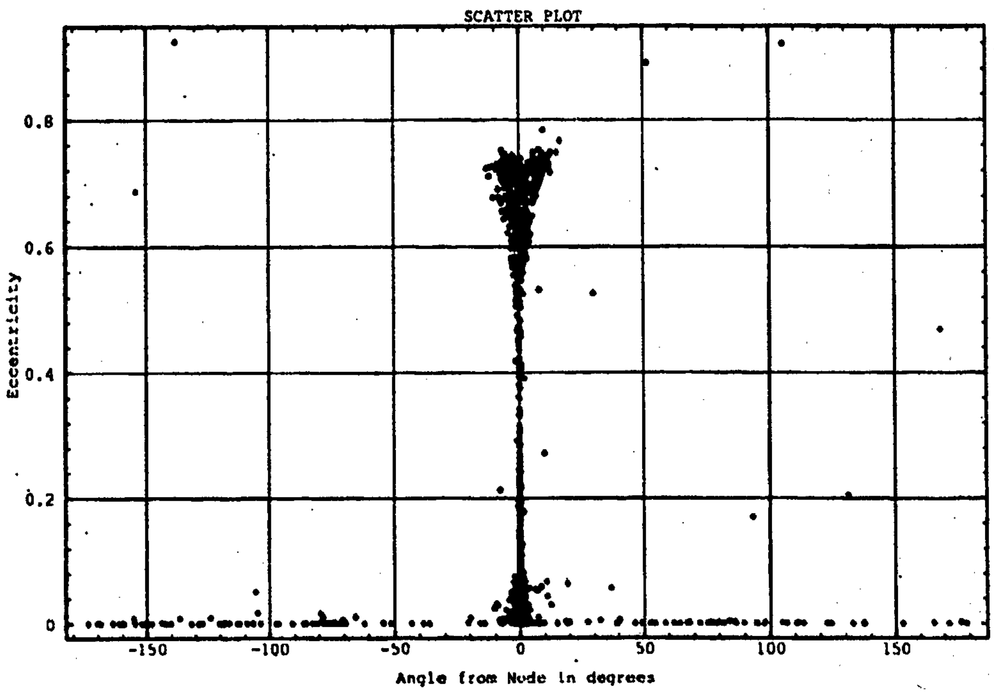


Figure 5.11. Scatter Plot -- Angle from Node and Eccentricity.

When both sources of angular differences from the node are taken into account, there are very few satellites which are not at the ascending node.

Another interesting result is the relationship between eccentricity and the Kozai mean motion depicted in Figure 5.12. The triangular shape is due to the fact that for a particular orbit, the Kozai mean motion has a limited eccentricity, otherwise the orbit intersects the surface of the earth. Equation 5.2 (Figure 5.13) shows the relationship between the mean motion and eccentricity to have a closed orbit which does not intersect the earth's surface.

$$R_{perigee} = \left(\frac{\mu}{n^2}\right)^{1/3}(1-e) \tag{5.1}$$

implies,

$$e \leq 1 - \left(\frac{n^2}{\mu}\right)^{1/3} R_\oplus \tag{5.2}$$

Since SGP4 does not use the time derivatives of mean motion which are contained in the NORAD two-line elements, $\dot{n}/2$ and $\ddot{n}/6$, we were interested in finding if there was some way of deriving the values of $\dot{n}/2$ and $\ddot{n}/6$ from bstar (B^*) and possibly other elements. Figure 5.14 shows there is no functional relationship between B^*, $\dot{n}/2$, and $\ddot{n}/6$.[4] The fact there is no functional relationship was also confirmed by personal interview with a member of USSPACECOM/J3SOT (8). Without the hoped for functional relation between orbital elements to determine the time derivatives of the Kozai mean motion, we do not have any way of differentially correcting $\dot{n}/2$ and $\ddot{n}/6$.

The analysis in this section allowed easy examination of the current satellite constellation composition. We examined how the satellite population fell into the various Gabbard classes with most of them falling into the low earth classes 1-6. We also examined some of the correlations between the orbital elements. First we confirmed the majority of satellites were propagated to the ascending node crossing. We also showed the limits on eccentricity given a particular Kozai mean motion. And finally, we generated a plot showing there was no functional relationship between B^*, $\dot{n}/2$, and $\ddot{n}/6$.

[4]Appendix E contains the side views of the three-dimensional plot for those of us who have trouble visualizing three dimensions on a two-dimensional sheet of paper. Figures E.7, E.8, and E.9 are the relevant scatter plots.

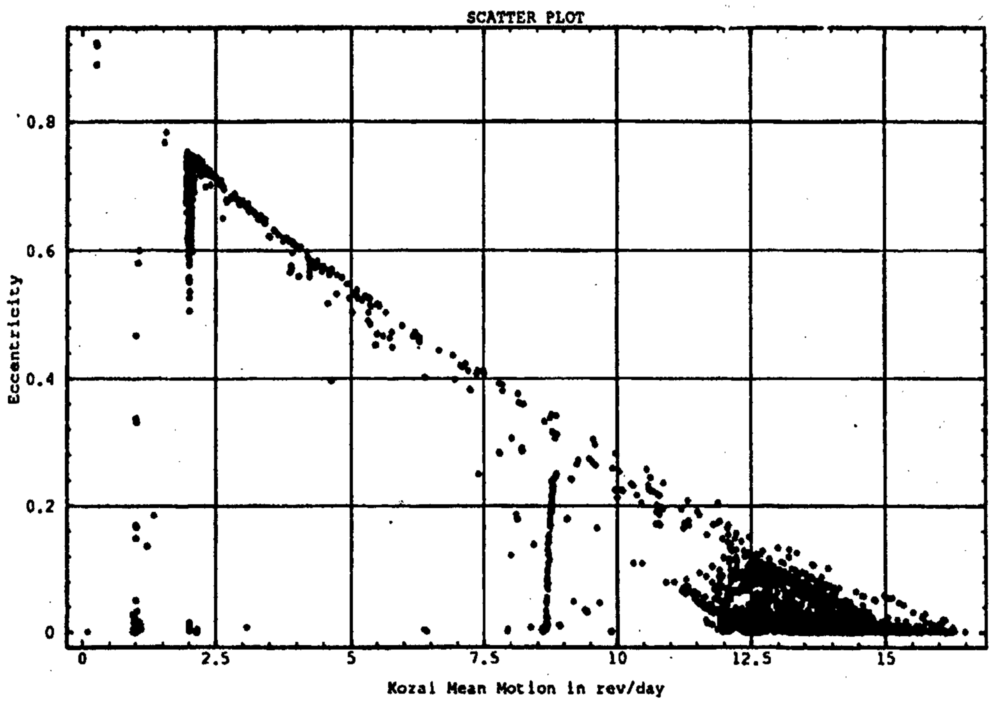


Figure 5.12. Scatter Plot — Kozai Mean Motion and Eccentricity.

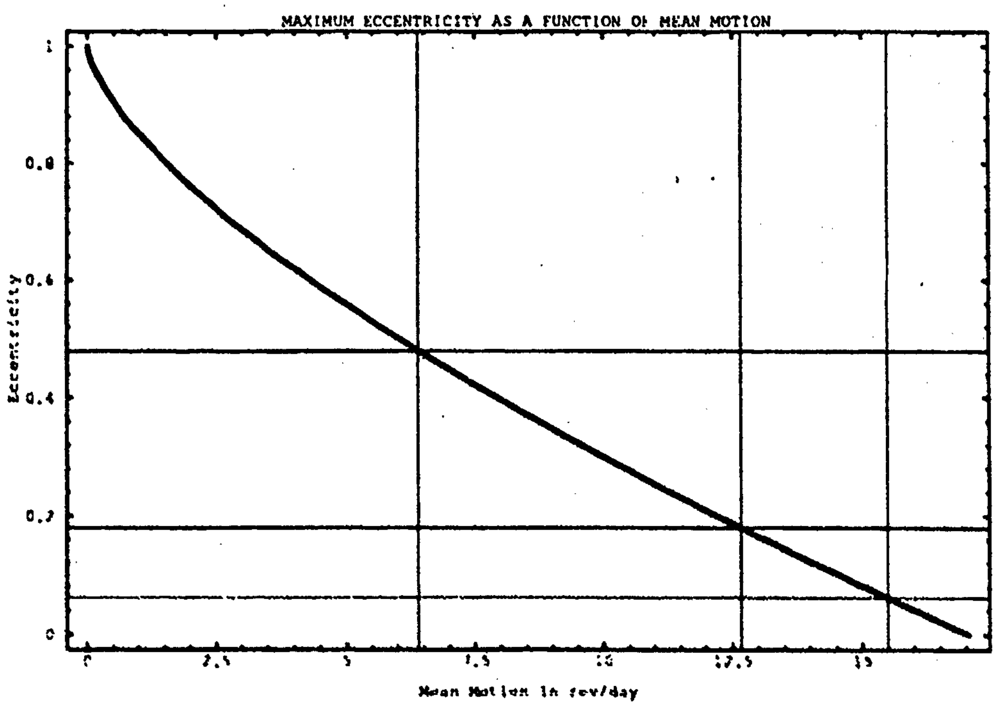


Figure 5.13. Maximum Eccentricity as a Function of Mean Motion.

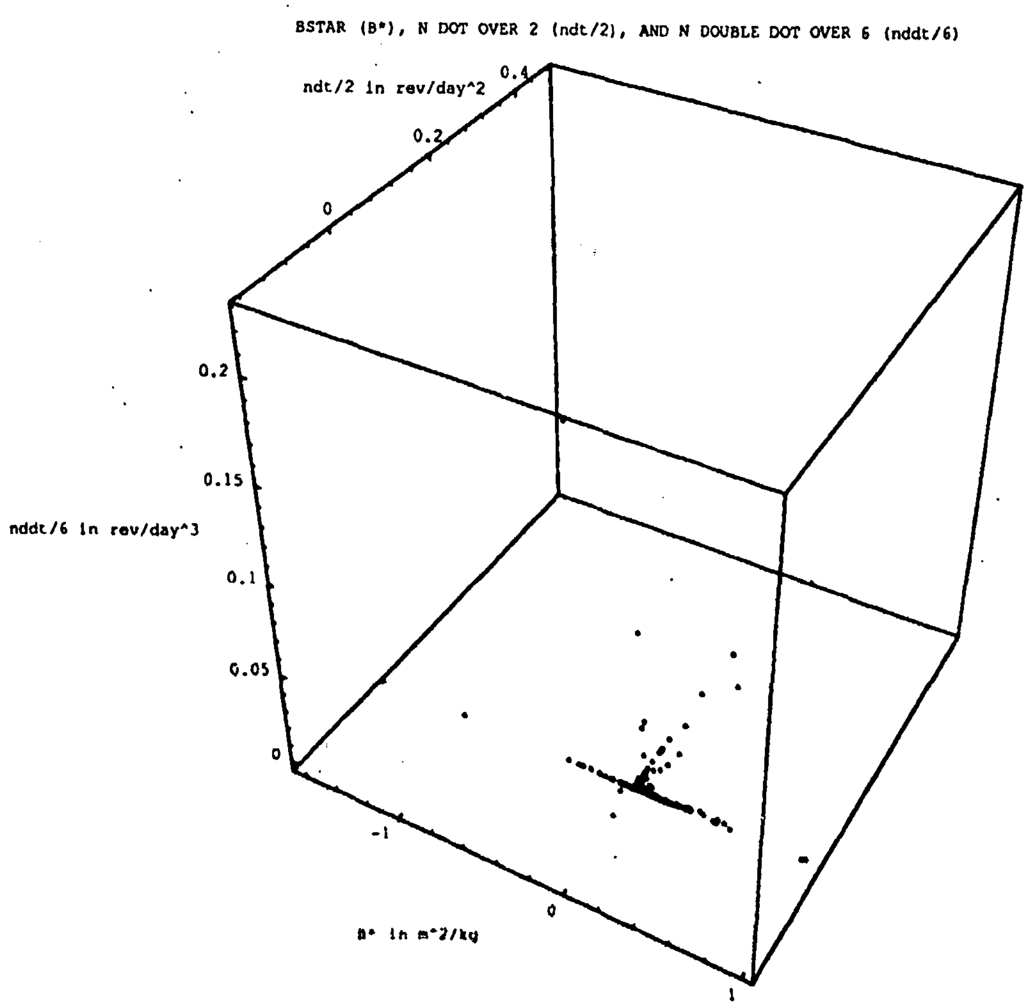


Figure 5.14. Scatter Plot B^*, $\dot{n}/2$, and $\ddot{n}/6$.

5.2 Perturbations Analysis

There are two reasons for looking at satellite perturbations, specifically the J_2 perturbations; to identify the magnitude of the perturbations due to J_2, and to aid in the propagation of orbital elements. In Section 3.1.1 we derived the actual equations which would be used in the propagattion algorithm within the differential corrector model. However, the examination of the magnitude of these perturbations is performed in this section.

Figure 5.15 shows the magnitude of the time derivative of the mean anomaly at epoch, $\dot{M}_0$, and this is directly the difference between the true mean motion and the Kozai mean motion. The "maximum" referred to in the graph uses the fact that the perturbations are largest when the eccentricity is at its maximum.[5] Figure 5.16 shows the differences in $\dot{M}_0$ between the maximum possible eccentricity and a zero eccentricity.[6]

Figure 5.17 shows the magnitude of the time derivative of the ascending node, $\dot{\Omega}$. Again as before, the "maximum" referred to in the graph uses the fact that the perturbations are largest when the eccentricity is at its maximum. Figure 5.18 shows the differences in $\dot{\Omega}$ between the maximum possible eccentricity and a zero eccentricity.

Figure 5.19 shows the magnitude of the time derivative of the argument of perigee $\dot{\omega}$. And for one last time, the "maximum" referred to in the graph uses the fact that the perturbations are largest when the eccentricity is at its maximum. Figure 5.20 shows the differences in $\dot{\omega}$ between the maximum possible eccentricity and a zero eccentricity.

The next four graphs, Figures 5.21 through 5.24, show the effects of varying one of n_0, e_0, or i_0, while holding the others constant. The perturbed elements for the following graphs are Ω_0 and ω_0. We consider M_0 an unperturbed element since for our purposes we do not really use the true mean motion but instead the Kozai mean motion. The graphs are all normalized to 1 at some meaningful point for the data. The perturbations

[5] The perturbation is a function of mean motion, inclination, and eccentricity. To make the perturbation "plot-able", we eliminated the eccentricity dependence. For each mean motion value a maximum eccentricity is computed using Equation 5.2 and the result is used in the perturbation equation, thus, eliminating the eccentricity dependence. The perturbation for a given mean motion and inclination is then plotted.

[6] To see the effects of eliminating the eccentricity dependence of the perturbation, the perturbations where also computed for a zero eccentricity instead of a maximum eccentricity. The zero eccentricity result was subtracted from the maximum eccentricity result showing eccentricity's impact on the perturbation for a given mean motion and inclination.

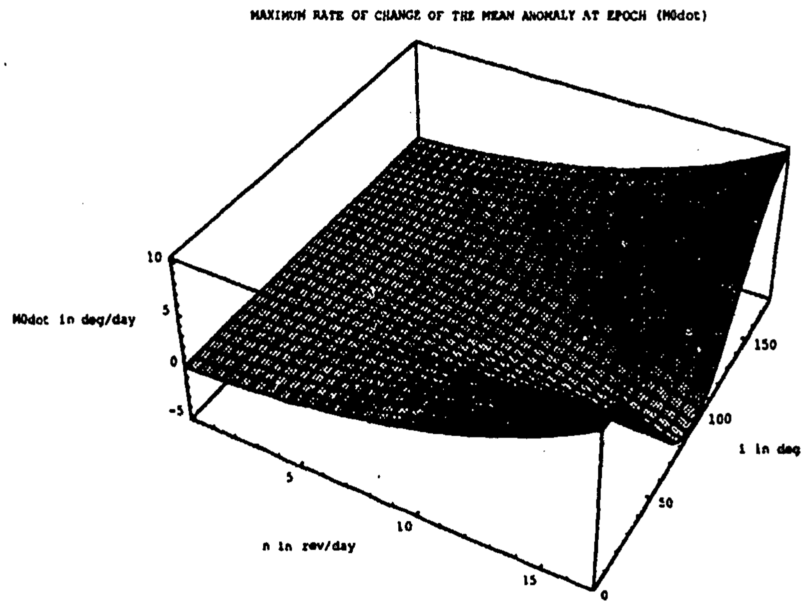


Figure 5.15. Perturbations on Mean Anomaly at Epoch.

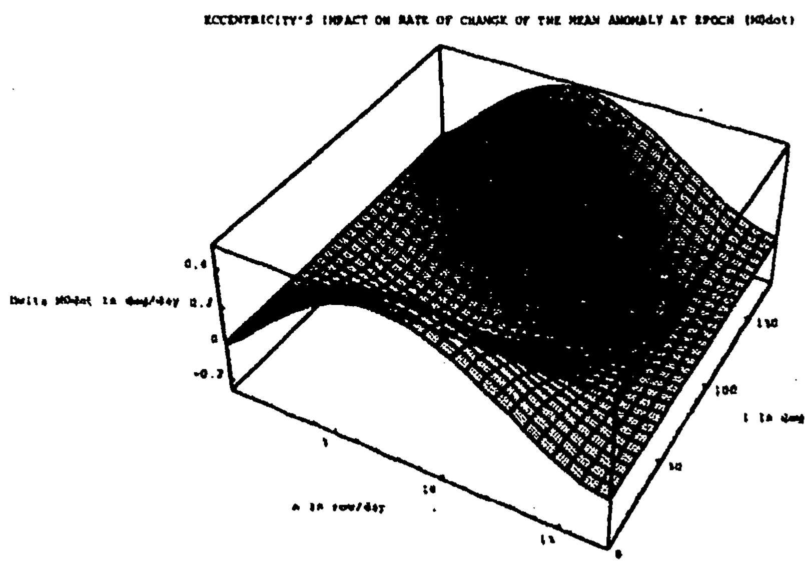


Figure 5.16. Eccentricity's Influence on Mean Anomaly at Epoch Perturbations.

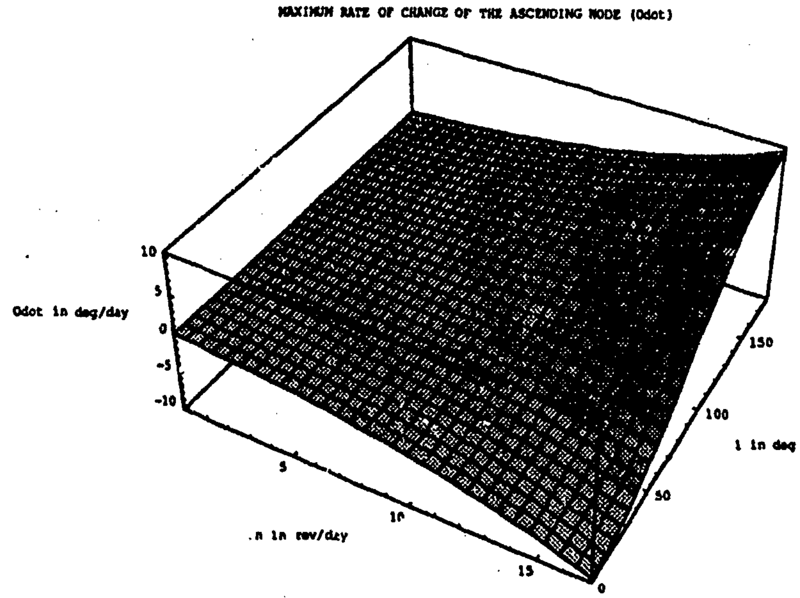


Figure 5.17. Perturbations on the Ascending Node.

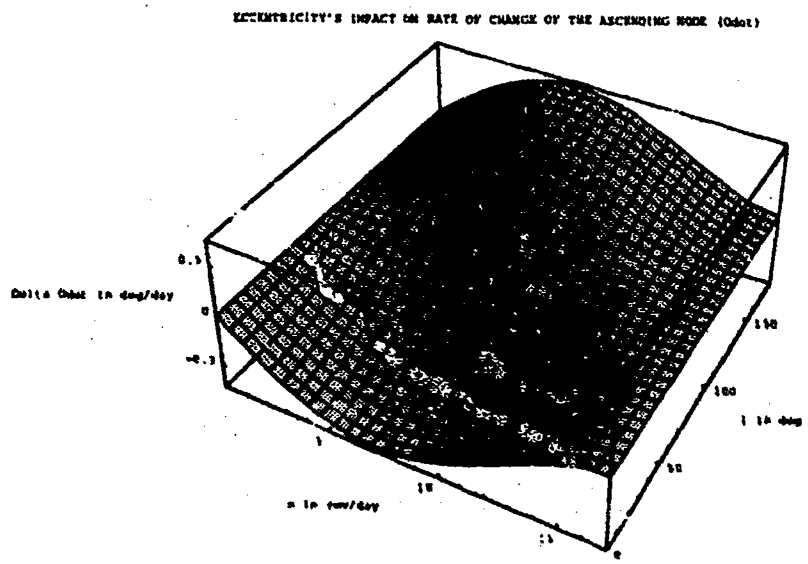


Figure 5.18. Eccentricity's Influence on Ascending Node Perturbations.

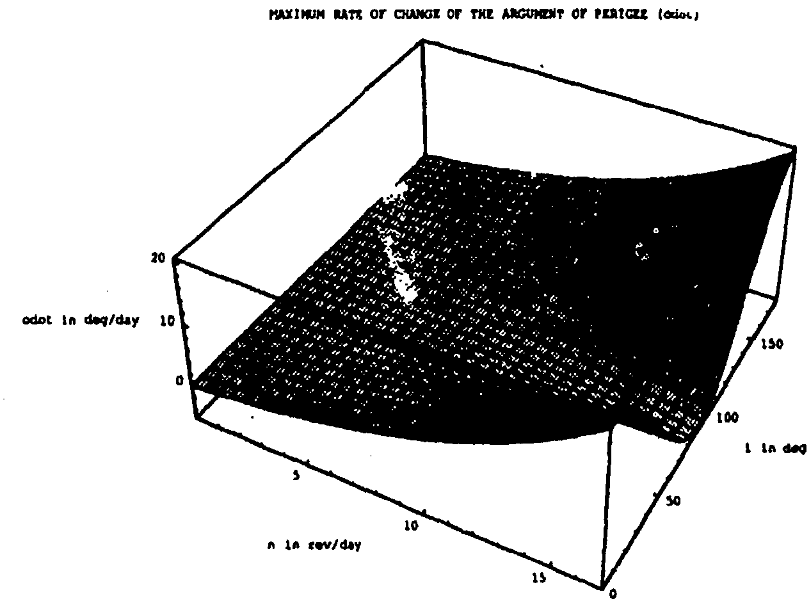


Figure 5.19. Perturbations on the Argument of Perigee.

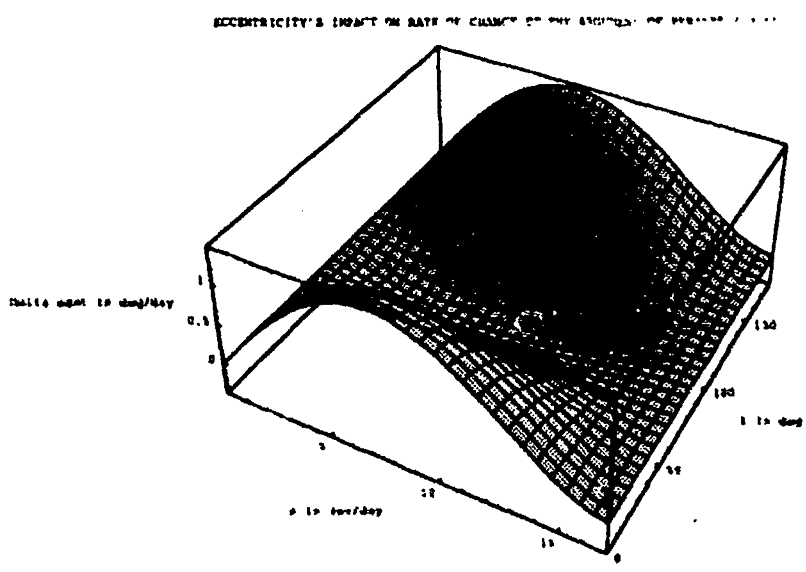


Figure 5.20. Eccentricity's Influence on Argument of Perigee Perturbations.

due to mean motion are set to 1 at the maximum possible mean motion. Eccentricity's perturbations are set to 1 at its minimum value since the effects "explode" at very high eccentricity. And finally, the perturbations caused by inclination are normalized to 1 at the maximum perturbation value.

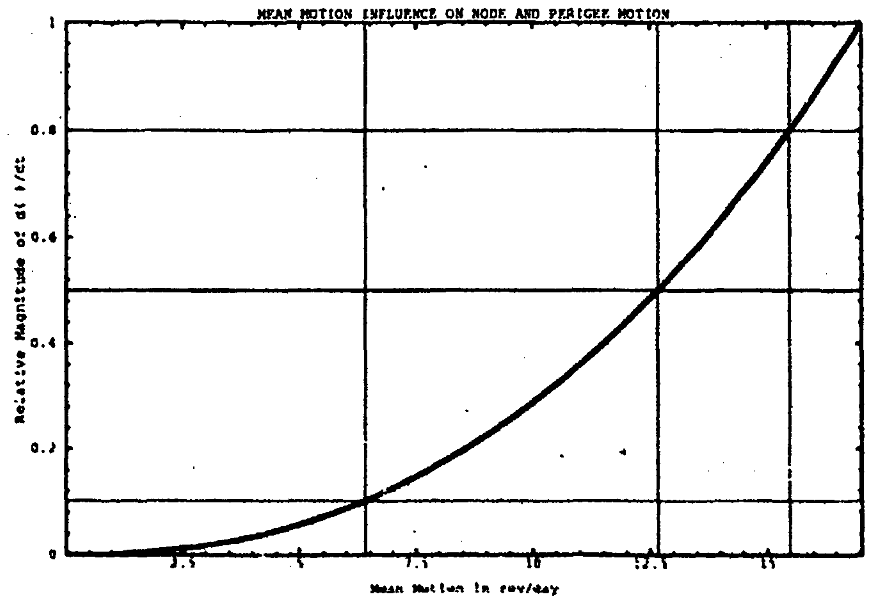


Figure 5.21. Mean Motion Influence on Node and Perigee Motion.

As can be seen by this section, the perturbation effects due to J_2 are very small, and eccentricity's impact is orders of magnitude less than the others since the mean motion's perturbation drops off faster than the eccentricity's perturbations can grow.

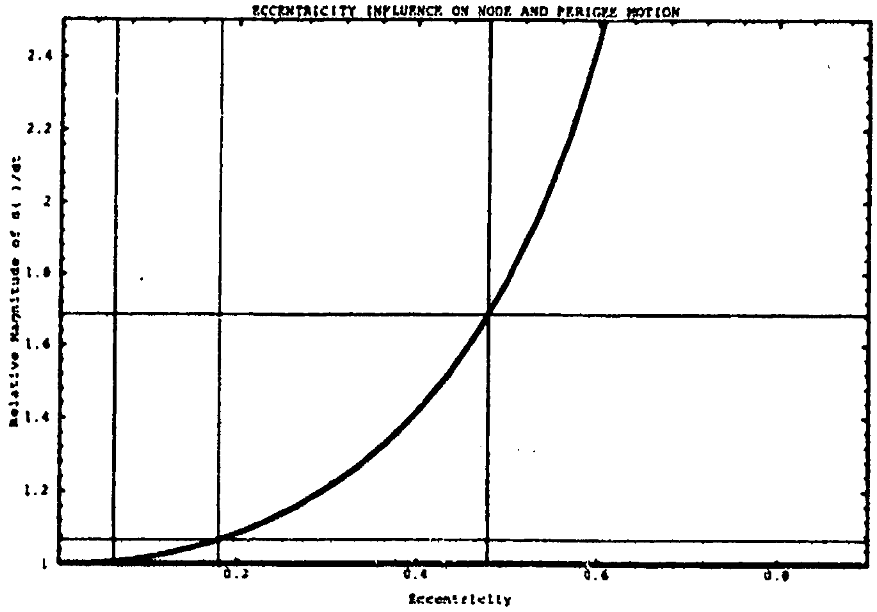


Figure 5.22. Eccentricity Influence on Node and Perigee Motion.

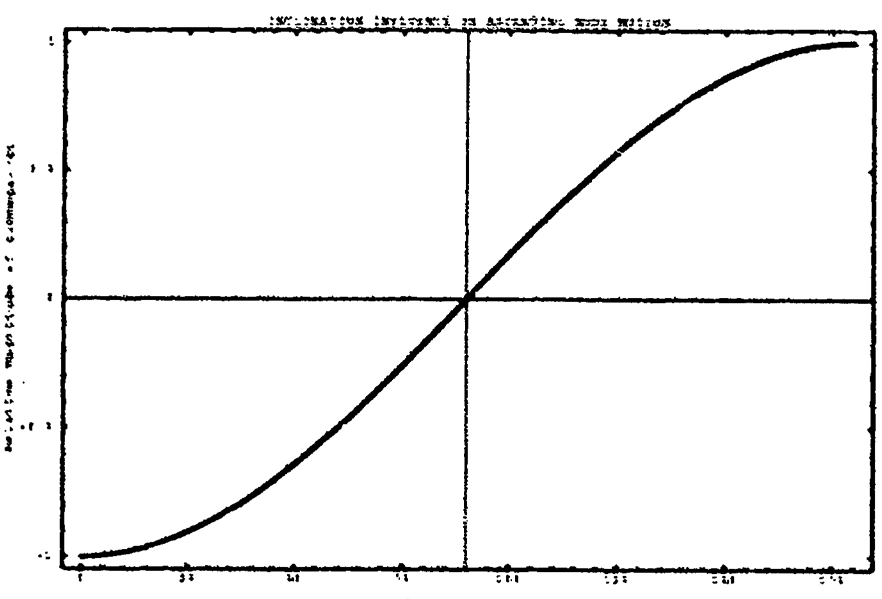


Figure 5.23. Inclination Influence on Node Motion.

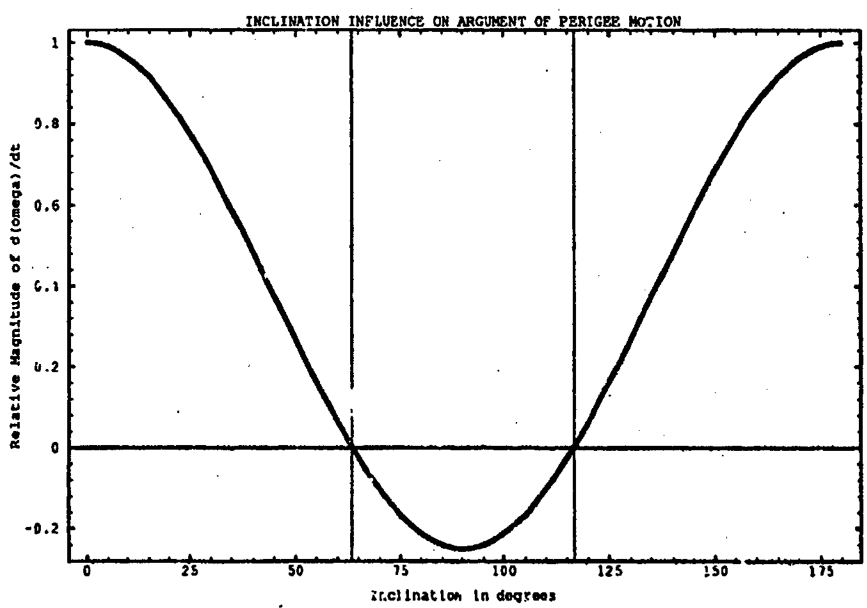


Figure 5.24. Inclination Influence on Perigee Motion.

5.3 Classification Scheme Development

The development of a orbital-element-based calssification scheme involved more of a statistical approach rather than an analysis of the current satellite population or concentration on the J_2 perturbations. After examining the distributions of satellites within the Gabbard classes and the impacts of the J_2 perturbations we decided to create our own classification scheme based on the perturbations the satellites experience. The population of satellites was divided into classes based on mean motion, eccentricity, and inclination since these are the elements which contribute to the perturbations. In the previous section the magnitudes and relative impact of $n_{\kappa,0}$, e_0 and i_0 on both the argument of perigee (ω_0) and the ascending node (Ω_0) were examined.

In the graphs in the previous section (Figures 5.21 through 5.24), some of the critical points were highlighted. For instance we considered inclination effects on $\dot{\Omega}$ and $\dot{\omega}$. There are three divisions which occur and Table 5.3 shows the directions of the perturbations.

Table 5.3. Sign Effects on Perturbations Due to Inclination.

Inclination	Sign	
	$\dot{\omega}$	$\dot{\Omega}$
degrees	+/−	+/−
$i < 63.4350$	+	−
$63.4350 < i < 90.0000$	−	−
$90.0000 < i < 116.5648$	−	+
$i > 116.5648$	+	+

Instead of using just the four inclination ranges corresponding to the four combinations of $\dot{\omega}$ and $\dot{\Omega}$ effects, we created two additional "deadbands" about the critical inclinations. The deadbands were picked to be a total width of 0.1 radians (about 6°) wide based on a closer inspection of the distribution of satellites near this inclination with higher eccentricities (Figure E.10). Table 5.4 summarizes the six inclination classes.

Besides inclination classes, we divided the mean motion into four classes. The number of classes was picked somewhat arbitrarily but with some logic behind the decision. The first class division we decided upon was the 225 minute period used to differentiate between the SGP4 and SDP4 orbit propagation models. We also created two other divisions; one

Table 5.4. Orbital Classes Based on Inclination.

Inclination Class	Inclination Min	Inclination Max	Sign $\dot{\omega}$	Sign $\dot{\Omega}$
	degrees	degrees	+/−	+/−
1	N/A	60.5702	+	−
2	60.5702	66.2998	0	−
3	66.2998	90.0000	−	−
4	90.0000	113.7000	−	+
5	113.7000	119.4296	0	+
6	119.4296	N/A	+	+

at the point where the perturbation level was 20 percent less than its maximum and the second at the 50 percent of maximum point. The grid-lines on Figure 5.21 in the previous section denote these divisions (see Table 5.5).

Table 5.5. Orbital Classes Based on Mean Motion.

Mean Motion Class	Mean Motion Min	Mean Motion Max
	rev/day	rev/day
1	N/A	6.4000
2	6.4000	12.6633
3	12.6633	15.4891
4	15.4891	N/A

Due to the previous analysis on J_2 perturbations, we decided let the eccentricity classes come from the mean motion classes. Figure 5.25 illustrates what was done. By using the maximum eccentricity based on mean motion, we were able to determine the maximum value for each of the three mean motion divisions. Figure 5.25 shows the three eccentricities which are at the intersections of the mean motion and maximum eccentricity line. These three eccentricities were then plotted on Figure 5.22 in Section 5.2. Since the low eccentricity value contributes very little to the perturbations, the low eccentricity value was dropped and the remaining two eccentricities were used as the dividing lines for eccentricity classes (see Table 5.6).

Table 5.7 defines each of the classes. To better illustrate the classifications Figure 5.26 shows the eccentricity and mean motion divisions. The six inclination classes occupy

Table 5.6. Orbital Classes Based on Eccentricity.

Eccentricity Class	Eccentricity	
	Min	Max
1	N/A	0.179665
2	0.179665	0.479510
3	0.479510	N/A

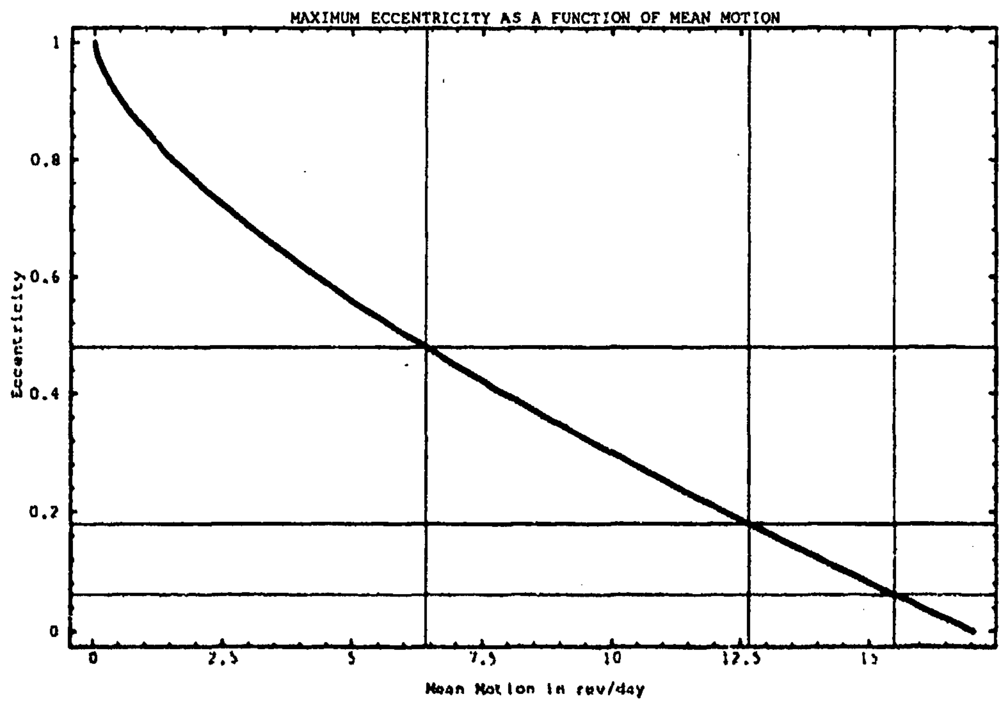


Figure 5.25. Maximum Eccentricity as a function of Mean Motion.

each of the seven rectangles in the lower-left corner. As a symbolic short hand, when a particular class is discussed it will be in the following order: mean motion – eccentricity – inclination. For example class 4-1-2 represents; Mean Motion Class 4, Eccentricity Class 1, and Inclination Class 2.

Table 5.7. Summary of Orbital-Element-Based Classes.

Class	Mean Motion		Eccentricity		Inclination	
	Min	Max	Min	Max	Min	Max
	rev/day	rev/day			degrees	degrees
1	N/A	6.4000	N/A	0.179665	N/A	60.5702
2	6.4000	12.6633	0.179665	0.479510	60.5702	66.2998
3	12.6633	15.4891	0.479510	N/A	66.2998	90.0000
4	15.4891	N/A			90.0000	113.7000
5					113.7000	119.4296
6					119.4296	N/A

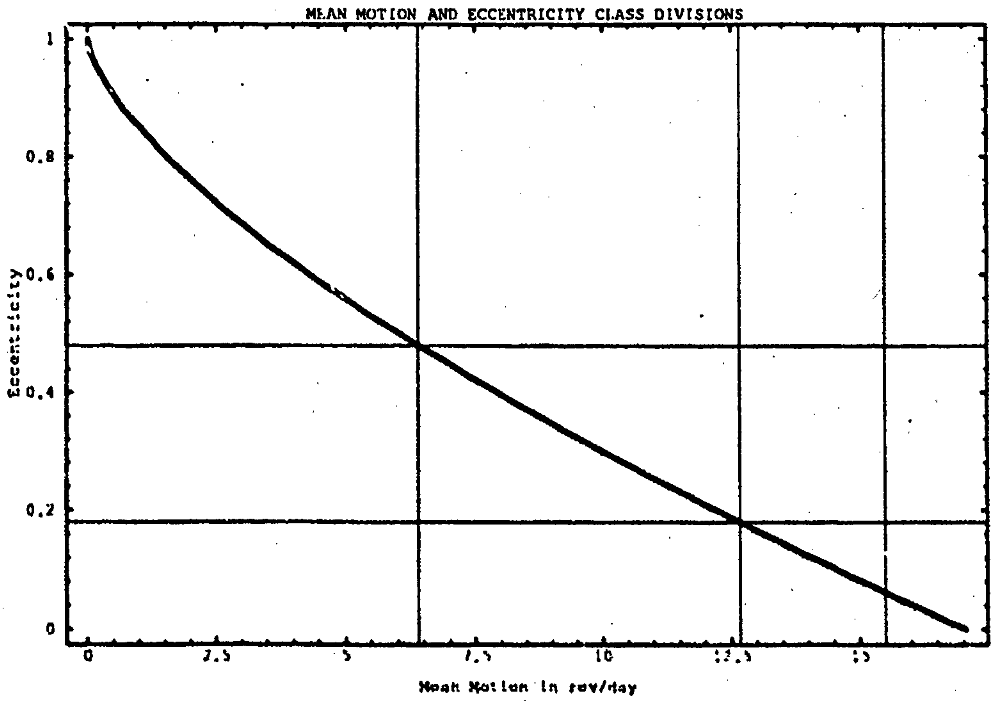


Figure 5.26. Mean Motion and Eccentricity Class Divisions.

With the classes described as in Table 5.7, the FORTRAN code for decoding the two-line orbital element sets was modified to also output the satellites into their respective

classes. Table 5.8 shows the number of satellite in each of the classes. Appendix F lists the FORTRAN source code used to divide the satellites into their classes. Appendix E.3 contains six graphs showing the distribution of the current population in these classes.

Table 5.8. Satellite Population Within Each Element-Based Class.

O E Class	Qty.	Pct.	Mean Motion Min	Mean Motion Max	Eccentricity Min	Eccentricity Max	Inclination Min	Inclination Max
		%	rev/day	rev/day	—	—	deg	deg
1-1-1	368	6.04	N/A	6.400	N/A	0.179	N/A	60.57
1-1-2	65	1.07	N/A	6.400	N/A	0.179	60.57	66.30
1-1-4	2	0.03	N/A	6.400	N/A	0.179	90.00	113.70
1-1-6	1	0.02	N/A	6.400	N/A	0.179	119.43	N/A
1-2-1	14	0.23	N/A	6.400	0.179	0.480	N/A	60.57
1-2-2	5	0.08	N/A	6.400	0.179	0.480	60.57	66.30
1-2-4	2	0.03	N/A	6.400	0.179	0.480	90.00	113.70
1-3-1	192	3.15	N/A	6.400	0.480	N/A	N/A	60.57
1-3-2	249	4.09	N/A	6.400	0.480	N/A	60.57	66.30
1-3-3	74	1.21	N/A	6.400	0.480	N/A	66.30	90.00
2-1-1	73	1.20	6.400	12.66	N/A	0.179	N/A	60.57
2-1-2	119	1.95	6.400	12.66	N/A	0.179	60.57	66.30
2-1-3	647	10.62	6.400	12.66	N/A	0.179	66.30	90.00
2-1-4	458	7.96	6.400	12.66	N/A	0.179	90.00	113.70
2-1-6	4	0.07	6.400	12.66	N/A	0.179	119.43	N/A
2-2-1	43	0.71	6.400	12.66	0.179	N/A	N/A	60.57
2-2-2	20	0.33	6.400	12.66	0.179	N/A	60.57	66.30
2-2-3	46	0.76	6.400	12.66	0.179	N/A	66.30	90.00
2-2-4	8	0.13	6.400	12.66	0.179	N/A	90.00	113.70
2-2-6	2	0.03	6.400	12.66	0.179	N/A	119.43	N/A
3-1-1	302	4.96	12.66	15.49	N/A	N/A	N/A	60.57
3-1-2	571	9.37	12.66	15.49	N/A	N/A	60.57	66.30
3-1-3	1631	26.77	12.66	15.49	N/A	N/A	66.30	90.00
3-1-4	1111	18.24	12.66	15.49	N/A	N/A	90.00	113.70
3-1-5	4	0.07	12.66	15.49	N/A	N/A	113.70	119.43
3-1-6	11	0.18	12.66	15.49	N/A	N/A	119.43	N/A
4-1-1	25	0.41	15.49	N/A	N/A	N/A	N/A	60.57
4-1-2	26	0.43	15.49	N/A	N/A	N/A	60.57	66.30
4-1-3	13	0.21	15.49	N/A	N/A	N/A	66.30	90.00
4-1-4	6	0.10	15.49	N/A	N/A	N/A	90.00	113.70
OTHERS	0							
ALL	6092							

5.4 *Representative Satellite Selection*

In order to pick a representative satellite from each of the newly formed classes, a Mathematica function was written to read in an entire class of satellites and then compute which satellite came closest to the 'average satellite' for the class. We decided on using this approach over using the average because we wanted to deal with actual satellites instead of a non-existent average satellite. Not all elements were used in determining the closest satellite to the 'average satellite.' The mean anomaly at epoch was not included. The reason for excluding the mean anomaly was we would expect the average to be very close to 180 degrees and which would make it very likely that satellites with a mean anomaly of 180 would be selected. Since mean anomaly changes constantly throughout the orbit it was an unnecessary restraint. Appendix F has the developed Mathematica function for performing the satellite selection, and it also lists the satellites selected in each of the classes.

Once a satellite was selected for each of the classes, we decided to hand pick a representative sample of classes (satellites) to perform our analysis on the differential corrector model. Table 5.9 shows the classes and the satellites picked. Three factors were part of the decision process at this point. First, we wanted to select classes which had particular types of satellites. For instance, Class 1 1 1 contains geosynchronous satellites. Second, the classes with the most satellites were selected for further analysis. Classes 3 1 3 and 3 1 4 were selected for this reason. And finally, some classes were selected for added diversity in our analysis. Classes 2 2 3 is an example of a class selected for added diversity.

Table 5.9. Classes Selected for Differential Corrector Model Analysis.

Class	Catalog Number	Class	Catalog Number
1 1 1	15141	3 1 1	01996
1 1 2	15259	3 1 2	14443
1 3 2	14199	3 1 3	19643
2 1 3	10293	3 1 4	17429
2 1 4	10393	4 1 1	20335
2 2 3	19859	4 1 2	15584

5.5 *Data Discrepancies*

Some problems were encountered in producing truth model data for the representative satellites selected from the mean motion range four classes (4-1-1, 4-1-2, 4-1-3, 4-1-4). Consequently, an individual cross check of each of the 67 satellites in these four classes with the Satellite Catalog (33) was performed. It was discovered that 46 of the 67 satellites in these four classes were listed as having decayed within 60 days of the orbital element set epoch data. As our testing criteria required a run of 60 days, these satellites were excluded from selection.

During the course of this inspection, we discovered 12 additional satellites included in these classes were listed as having decayed *prior to* the epoch data of the orbital element set. These satellites, and the pertinent data, are included in Table 5.10.

Table 5.10. Orbital Element Set Discrepancies.

Satellite Number	Element Set Number	Element Set Epoch Date	Satellite Catalog Decay Date
13222	001	90111.81766866	14 Aug 92
16640	000	90110.75000000	05 Apr 86
02673	000	90101.75891899	13 Dec 67
13078	001	90116.31294873	06 Apr 82
13102	000	90117.80489262	01 Apr 82
13179	000	90118.60021568	19 May 82
19131	073	90086.49108799	31 May 88
20334	236	90061.55933732	21 Jan 90
13174	000	90107.44919305	15 May 82
01073	000	90063.49859744	12 Nov 64
01473	000	90081.29833724	13 Mar 69
13097	000	90118.78672990	21 Mar 82

Compiled from (33)

VI. ANALYSIS OF DIFFERENTIAL CORRECTOR MODEL

As discussed in Chapter IV, we divided the satellite population into 42 classes based on combinations of six inclination, three eccentricity, and four mean motion ranges.[1] Based on the analysis of the satellite population discussed in Chapter VI, we divided the satellite catalogue into the 33 populated classes and picked the "average" satellite from each class. These satellites are listed in Appendix F.6. From these 33 classes, we selected 12 satellites to analyze. These satellites are listed in the table below.

Table 6.1. Representative Satellites for Analysis.

Class	Catalog Number	Class	Catalog Number
1-1-1	15141	3-1-1	01996
1-1-2	15259	3-1-2	14443
1-3-2	14199	3-1-3	19643
2-1-3	10293	3-1-4	17429
2-1-4	10393	4-1-1	20335
2-2-3	19859	4-1-2	15584

Using the methodology discussed in Chapter IV, an orbital element set for each of these 12 satellites was processed through the truth model to obtain simulated sensor measurements. The model created a total of 40 random observation files; 10 random observation files for each of the four observation rates (two, four, six, and eight) to be tested. Each random observation file was then processed through DIFC for each of the four LUPIs (two, four, six, and eight days) to be tested. The first-pass and last-pass VMAGT errors (residuals) for each random observation file were calculated and stored to the appropriate files based on LUPI/OPD combination. As discussed in Chapter IV, a fixed mean motion variance limit of 10^{-15} rad^2/min^2 was used.

6.1 *Analysis of Individual Satellite VMAGT Data.*

For each satellite/LUPI/OPD combination, all data from the 10 VMAGT files were plotted in conjunction with a four-day moving average using the Mathematica software plot

[1] The difference between 42 classes and 6×3×4 = 72 is that 30 of the combinations of i_0, e_0, and n_0 are impossible.

routine. Using the programming ability built into Mathematica, we constructed a program to calculate the 95 percent confidence intervals of the means, variances and 99 percent confidence levels as presented in Equations 4.27, 4.28, and 4.29. (Reference Appendix G for the Mathematica code used to perform these calculations and produce the graphs. Reference Appendix H for the VMAGT graphs and summary tables for each of the 16 LUPI/OPD combinations tested for the 12 satellites in Table 6.1.)

The following sections review each sample satellite tested, examining the data with respect to the following four performance measures:

1. "First-pass" VMAGT data. Expect to observe reasonable values (approximately 14 km or less for near-earth satellites and 20 to 40 km for deep-space satellites), a "saw-tooth" correction pattern, and no trends or serious outlying data.
2. "Last-pass" VMAGT values. Expect to observe steady-state, randomly-distributed values at a level less than first-pass values.
3. OPD effect on the 99 percent CL. Expect to observe that 99 percent CL value decreases as OPD increases.
4. LUPI effect on the 99 percent CL. Expect to observe that 99 percent CL value increases as LUPI increases.
5. Analysis of variance (ANOVA). Expect this analysis to verify statistical differences exist in 99 percent CL value results based on effects of OPD and LUPI combinations.

6.1.1 Class 1 1 1 (NORAD Catalog Number 15141). A summary of the confidence interval analysis and the VMAGT graphs for all LUPI/OPD combinations tested for this satellite are located in Appendix H.1. Data analysis with respect to the performance measures above indicate the following:

- Examination of the first-pass VMAGT values for this satellite indicated two peaks. The peaks for residual values occurred in the periods of Days 2 through 10 and Days 36 through 46. We believe this indicates a problem with the differential corrector program correcting to, and propagating, a bad state estimate. This could be

caused by the mean motion variance limit being unnecessarily large, thus reducing the confidence in a previously good estimate.

- Examination of the last-pass VMAGT values showed randomly-distributed, steady-state values at a level less than first-pass values.

- Examination of the OPD effects on the 99 percent CL show that, for all cases with this satellite, the 99 percent CL decreases as OPD increases. This was the expected result.

- Examination of the LUPI effects on the 99 percent CL show that, for the OPD 2 cases with this satellite, the 99 percent CL increases as LUPI increases. This was the expected result. However, for the OPD 4, 6, and 8 cases, the large overlapping confidence intervals make a determination impossible.

- ANOVA indicates there are differences in results based on OPD and LUPI. However, there appears to be no difference in results based on the interaction of LUPI and OPD.

6.1.2 Class I 1 2 (NORAD Catalog Number 15259). A summary of the confidence interval analysis and the VMAGT graphs for all LUPI/OPD combinations tested for this satellite are located in Appendix H.2. Data analysis with respect to the performance measures above indicate the following:

- Examination of the first-pass VMAGT values for this satellite indicated reasonable and consistent data with expected results. However, with the data from LUPI 8, there is a noticeable peak in the data during Batch 5. This effect may be due to the DC producing a bad estimate at the end of Batch 4. Additionally, the 99 percent CL for the best possible LUPI/OPD combination (8/2) was well below the 14 km VMAG limit used by USSPACECOM.

- Examination of the last-pass VMAGT values showed randomly-distributed, steady-state values at a level less than first-pass values.

- Examination of the OPD effects on the 99 percent CL show that the 99 percent CL generally decreases as OPD increases. OPD 4 for LUPI 2 and 4 increased slightly but decreased for the 6 and 8 OPD. This decreasing trend was the expected result.
- Examination of the LUPI effects on the 99 percent CL show that, for OPD 2 cases with this satellite, the 99 percent CL increases as LUPI increases. This was the expected result. However, for the OPD 4, 6, and 8 cases, the large overlapping confidence intervals make a determination impossible.
- ANOVA indicates there are differences in results based on OPD and LUPI. Additionally, there appears to be a difference in results based on the interaction of LUPI and OPD.

6.1.3 Class 1 3 2 (NORAD Catalog Number 14199). A summary of the confidence interval analysis and the VMAGT graphs for all LUPI/OPD combinations tested for this satellite are located in Appendix H.3. Data analysis with respect to the performance measures above indicate the following:

- Examination of the first-pass VMAGT values for this satellite showed large values with occasional peaks with a slight positive slope of the data within batches. The large VMAGT values are probably due to the properties associated with the orbital dynamics associated with this class.
- Examination of the last-pass VMAGT values showed randomly-distributed, steady-state values at a level less than first-pass values.
- Examination of the OPD effects on the 99 percent CL show that, for all cases with this satellite, the 99 percent CL decreases as OPD increases. This was the expected result.
- Examination of the LUPI effects on the 99 percent CL show that, for the OPD 2 cases, the 99 percent CL increases as LUPI increases. For all other cases, the confidence intervals are very large and overlapping, and the 99 percent CL are not significantly different enough to draw any conclusions.

- ANOVA indicates there are differences in results based on OPD, but not on LUPI. Additionally, there appears to be no difference in results based on the interaction of LUPI and OPD.

6.1.4 Class 2-1-3 (NORAD Catalog Number 10293). A summary of the confidence interval analysis and the VMAGT graphs for all LUPI/OPD combinations tested for this satellite are located in Appendix H.4. Data analysis with respect to the performance measures above indicate the following:

- Examination of the first-pass VMAGT values for this satellite showed small values which showed occasional peaks and a slight positive slope. The 99 percent CL for the best possible LUPI/OPD combination (8/2) was well below the 14 km VMAG limit used by USSPACECOM.
- Examination of the last-pass VMAGT values showed randomly-distributed, steady-state values at a level less than first-pass values with a slight periodic effect.
- Examination of the OPD effects on the 99 percent CL show that, for all cases with this satellite, the 99 percent CL decreases as OPD increases. This was the expected result.
- Examination of the LUPI effects on the 99 percent CL show that, for all cases with this satellite, the overlap of the 99 percent CL makes a determination impossible for this satellite. We believe this situation could be eliminated in future research by increasing the number of random observation runs.
- ANOVA indicates there are differences in results based on OPD and LUPI. There may be a difference in results based on the interaction of LUPI and OPD. However, the ANOVA results are the same to the measured accuracy and no determination can be made.

6.1.5 Class 2-1-4 (NORAD Catalog Number 10393). A summary of the confidence interval analysis and the VMAGT graphs for all LUPI/OPD combinations tested for this satellite are located in Appendix H.5. Data analysis with respect to the performance measures above indicate the following:

- Examination of the first-pass VMAGT values for this satellite indicated small, reasonable, and consistent values with a slight positive slope. However, the 99 percent CL for the best possible LUPI/OPD combination (8/2) was well below the 14 km VMAG limit used by USSPACECOM.

- Examination of the last-pass VMAGT values showed randomly-distributed, steady-state values at a level less than first-pass values.

- Examination of the OPD effects on the 99 percent CL show that, for all cases with this satellite, the 99 percent CL decreases as OPD increases. This was the expected result.

- Examination of the LUPI effects on the 99 percent CL show that, for all cases with this satellite, the confidence interval overlap of the 99 percent CL makes a determination impossible for this satellite. We believe this situation could be eliminated in future research by increasing the number of random observation runs.

- ANOVA indicates there are differences in results based on OPD and LUPI. However, there is not a difference in results based on the interaction of LUPI and OPD.

6.1.6 Class 2-2-3 (NORAD Catalog Number 19859). A summary of the confidence interval analysis and the VMAGT graphs for all LUPI/OPD combinations tested for this satellite are located in Appendix H.6. Data analysis with respect to the performance measures above indicate the following:

- Examination of the first-pass VMAGT values for this satellite showed small, reasonable, and consistent values with a slight positive slope. The 99 percent CL for the best LUPI/OPD combination (8/2) was well below the 14 km VMAG limit used by USSPACECOM.

- Examination of the last-pass VMAGT values showed steady-state values at a level less than first-pass values. The data also showed a slight periodic effect, damping with each successive batch correction.

- Examination of the OPD effects on the 99 percent CL show that, for all cases with this satellite, the 99 percent CL decreases as OPD increases. This was the expected result.

- Examination of the LUPI effects on the 99 percent CL show that, for all cases with this satellite, the overlap of the 99 percent CL makes a determination impossible for this satellite. We believe this situation could be eliminated in future research by increasing the number of random observation runs.

- ANOVA indicates there are differences in results based on OPD but not on LUPI. However, there is a difference in results based on the interaction of LUPI and OPD.

6.1.7 Class 3-1-1 (NORAD Catalog Number 01996). A summary of the confidence interval analysis and the VMAGT graphs for all LUPI/OPD combinations tested for this satellite are located in Appendix H.7. Data analysis with respect to the performance measures above indicate the following:

- Examination of the first-pass VMAGT values for this satellite showed a severe positive slope on all batches. This trend becomes noticeable on the LUPI 4 combinations and is extremely obvious on the LUPI 8 combinations. Reference Figure H.62 for a good example of this trend. This trend appears to indicate that the covariance has become too exact (very small) and therefore the differential corrector is not allowing the elements to be corrected. Despite this apparent problem with the DC, the 99 percent CL for all LUPI 2 combinations was below the 14 km VMAG limit used by USSPACECOM.

- Examination of the last-pass VMAGT values showed values less than first-pass values. However, the data showed a slight periodic effect with peaks at the beginning and end of each batch.

- Examination of the OPD effects on the 99 percent CL show that, for all cases except the LUPI 2, OPD 8 combination, the 99 percent CL decreases as OPD increases. In the LUPI 2, OPD 8 combination, the large overlap of the confidence intervals makes a determination impossible. These results are consistent with those expected.

- Examination of the LUPI effects on the 99 percent CL show that, for all cases with this satellite, the 99 percent CL increases as LUPI increases. This was the expected result.

- ANOVA indicates there are differences in results based on OPD and LUPI. Additionally, there is a difference in results based on the interaction of LUPI and OPD.

6.1.8 Class 3-1-2 (NORAD Catalog Number 14443). A summary of the confidence interval analysis and the VMAGT graphs for all LUPI/OPD combinations tested for this satellite are located in Appendix H.8. Data analysis with respect to the performance measures above indicate the following:

- Examination of the first-pass VMAGT values for this satellite showed a slight positive slope on all batches. Despite this apparent problem with the DC, the 99 percent CL for all LUPI/OPD combinations with LUPI of 2, 4, and 6, was below 14 km VMAG limit used by USSPACECOM.

- Examination of the last-pass VMAGT values showed values less than first-pass values. However, the data showed a periodic effect with an apparent slight upward slope of the data within each batch

- Examination of the OPD effects on the 99 percent CL show that, for all cases except all OPD 8 combinations, the 99 percent CL decreases as OPD increases. In all OPD 8 combinations, the large overlap of the confidence interval with the respective OPD 6 confidence interval makes a determination impossible. These results are consistent with those expected.

- Examination of the LUPI effects on the 99 percent CL show that, for all cases with this satellite, the 99 percent CL increases as LUPI increases. This was the expected result.

- ANOVA indicates there are differences in results based on OPD and LUPI. However, there is not a difference in results based on the interaction of LUPI and OPD.

6.1.9 Class 3-1-3 (NORAD Catalog Number 19643). A summary of the confidence interval analysis and the VMAGT graphs for all LUPI/OPD combinations tested for this satellite are located in Appendix H.9. Data analysis with respect to the performance measures above indicate the following:

- Examination of the first-pass VMAGT values for this satellite showed an positive slope on some batches. This slope is noticeable on the LUPI 6 combinations and extremely obvious on the LUPI 8 combinations. Reference Figure H.80 for a good example of this trend. This trend appears to indicate that the covariance has become too exact (very small) and therefore the differential corrector is not allowing the elements to be corrected. Despite this apparent problem with the DC, the 99 percent CL for some LUPI/OPD combinations was below 14 km VMAG limit used by USSPACECOM. The best LUPI/OPD combinations performing better than the USSPACECOM limit are the 6/4 (best LUPI) and the 4/2 (best OPD) combinations.

- Examination of the last-pass VMAGT values showed randomly-distributed, steady-state values at a level less than first-pass values. This was the expected result.

- Examination of the OPD effects on the 99 percent CL show that, for all cases except LUPI 2, OPD 4, and LUPI 2, OPD 6 combinations, the 99 percent CL decreases as OPD increases. In these two cases the large overlap of the confidence intervals makes a determination impossible. These results are consistent with those expected.

- Examination of the LUPI effects on the 99 percent CL show that, for all cases with this satellite, the 99 percent CL increases as LUPI increases. This was the expected result.

- ANOVA indicates there are differences in results based on OPD and LUPI. However, there is not a difference in results based on the interaction of LUPI and OPD.

6.1.10 Class 3-1-4 (NORAD Catalog Number 17429). A summary of the confidence interval analysis and the VMAGT graphs for all LUPI/OPD combinations tested for this satellite are located in Appendix H.10. Data analysis with respect to the performance measures above indicate the following:

- Examination of the first-pass VMAGT values for this satellite, especially the longer LUPIs, showed a similar upward sloping trend on some batches. This trend typically alternates magnitude (high–low) between batches. A good example of this is seen in Figure H.87. It appears to indicate that the covariance has become too exact (very small) and therefore the differential corrector is not allowing the elements to be corrected.
- Examination of the last-pass VMAGT values showed values less than first-pass values. However, a slight upward trend from batch to batch was noted, with later batches having growing ranges of values.
- Examination of the OPD effects on the 99 percent CL show that, for all cases with this satellite, the 99 percent CL decreases as OPD increases. This was the expected result.
- Examination of the LUPI effects on the 99 percent CL show that, for all cases with this satellite, the 99 percent CL increases as LUPI increases. This was the expected result.
- ANOVA indicates there are differences in results based on LUPI but not on OPD. Additionally, there is not a difference in results based on the interaction of LUPI and OPD.

6.1.11 Class 4 1 1 (NORAD Catalog Number 20355). A summary of the confidence interval analysis and the VMAGT graphs for all LUPI/OPD combinations tested for this satellite are located in Appendix H.11. Data analysis with respect to the performance measures above indicate the following:

- Examination of the first-pass VMAGT values for this satellite showed that the data was distributed over a wide range with steady-state mean values between 48 and 272 km. Examination of the OPD 6 and 8 graph (Figure H.92) shows a definite upward slope. This, like many previous satellites analyzed, seems to indicate that the covariance has become too exact (very small) and therefore the differential corrector is not allowing the elements to be corrected.

- Examination of the last-pass VMAGT values showed randomly-distributed, steady-state values at a level less than first-pass values. They are, however, much larger than other last-pass values seen in the analysis of other satellites.

- Examination of the OPD effects on the 99 percent CL show that, for all cases with this satellite, the 99 percent CL decreases as OPD increases. This was the expected result.

- Due to program run time and extremely large VMAGT values, only the LUPI two combinations were analyzed for this satellite. Therefore, no LUPI effects on the 99 percent CL could be noted.

- Due to the limited nature of the output, ANOVA was not conducted on this satellite. The absence of a complete output data set would make ANOVA comparisons unreliable.

6.1.12 Class 4 1 2 (NORAD Catalog Number 15584). A summary of the confidence interval analysis and the VMAGT graphs for all LUPI/OPD combinations tested for this satellite are located in Appendix H.12. Data analysis with respect to the performance measures above indicate the following:

- Examination of the first-pass VMAGT values for this satellite showed that the data was distributed over a wide range with steady-state mean values between 38 and 225 km. Examination of the OPD 6 and 8 graph (Figure H.95) shows a definite upward slope. This, like many previous satellites analyzed, seems to indicate that the covariance has become too exact (very small) and therefore the differential corrector is not allowing the elements to be corrected.

- Examination of the last-pass VMAGT values showed randomly-distributed, steady-state values at a level less than first-pass values. They are, however, much larger than other last-pass values seen in the analysis of other satellites.

- Examination of the OPD effects on the 99 percent CL show that, for all cases with this satellite, the 99 percent CL decreases as OPD increases. This was the expected result.

- Due to program run time and extremely large VMAGT values, only the LUPI two combinations were analyzed for this satellite. Therefore, no LUPI effects on the 99 percent CL could be noted.

- Due to the limited nature of the output, ANOVA was not conducted on this satellite. The absence of a complete output data set would make ANOVA comparisons unreliable.

6.2 *Tuning the Differential Corrector*

As mentioned in numerous analyses of the sample satellites in Section 6.1, our DC appears to have a problem with the correction of the elements. It appears from the VMAGT plots of many of the satellites tested that the covariance gets so small that it will not allow some, or all, of the elements to be corrected by the value that is actually needed. This would result in a "bad" estimate. and the the large VMAGT values seen in the above analysis.

This problem was first noted in the validation of the differential correction model. All satellites being tested exhibited the same severe positive slope to the VMAGT data within each batch. Figure 6.1 shows the data for Class 3-1-2 (Catalog Number 14443) as encountered during validation. This is representative of the data for all test cases.

We assumed this effect was due to a bad mean motion estimate. Through experimentation, we determined that a mean motion variance "limit" of 10^{-15} seemed to perform the best. During the experimentation, the "limit" appeared to be dependent on the batch size. However, no combination of batch size, LUPI, and/or OPD multiples could be determined which worked well. Therefore, the 10^{-15} limit was hard coded into the model such that if the (7,7) element ($\sigma^2_{n_o}$) of the covariance matrix was less than this value, then $\sigma^2_{n_o}$ was set to 10^{-15}.

The model was then used on all 16 LUPI/OPD combinations for all 12 satellite (orbit) classes investigated. As illustrated by the current LUPI 8, OPD 8, VMAGT plots for Class 3-1-2, Catalog Number 14443 (Figure II.71), this limitation appeared to work for most classes. However, this limit apparently did not work for Classes 3-1-1, 3-1-3, 3-1-4,

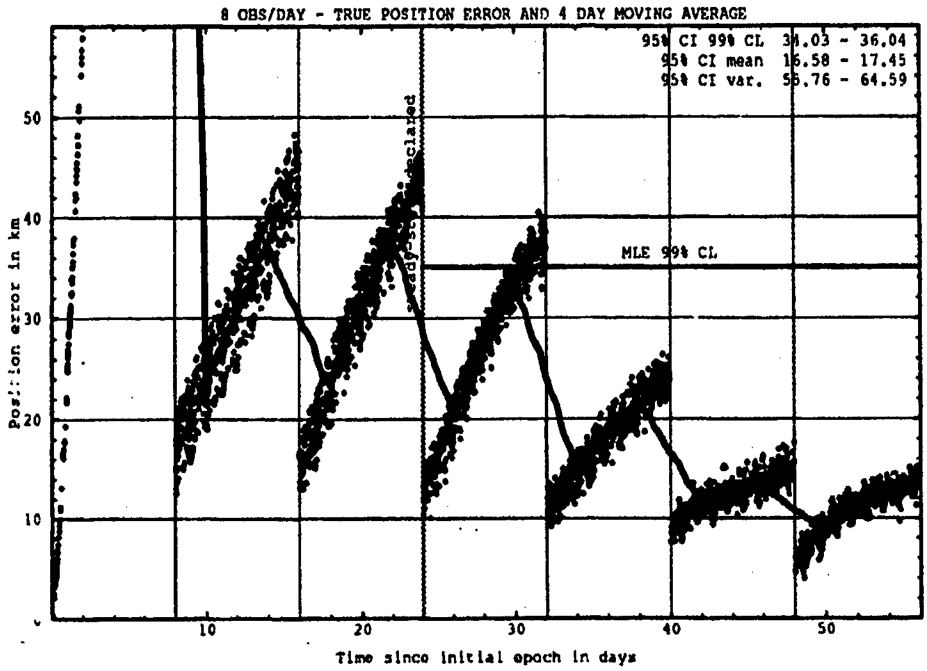


Figure 6.1. Old VMAGT data for Class 3 1-2 (Catalog Number 14443).

4 1 1, and 4-1-2 (Catalog Numbers 01996, 19643, 17429, and, possibly, 20335 and 15584) as shown in in Sections H.7, H.9, H.10, H.11, and H.12, respectively. We believed this indicated two things: first, the limit for these satellites should have been set at a higher value, and second, the limit value required is satellite, or orbit, dependent.

In attempt to determine if the mean motion variance limit used in our model was not restrictive enough, the five classes (sample orbits) exhibiting the severe positive slope of the VMAGT data were examined. Those classes (orbits) were:

3 1 1	(Catalog Number 01996)
3 1 3	(Catalog Number 19643)
3 1 4	(Catalog Number 17429)
4 1 1	(Catalog Number 20335)
4 1 2	(Catalog Number 15584)

Assuming that the VMAGT errors were primarily in-track errors (i.e., caused by a bad mean motion estimate), a first-order-of-magnitude estimate of the change in the mean motion was made. The slope of the VMAGT data gives an approximation of the rate of change of the in-track errors in km/day. The semi-major axis, a_0, is calculated as shown in Equation 3.14. If a circular orbit is assumed, the angular rate of change can be approximated from the the rate of change of the in-track error and the semi-major axis.

All six satellites have eccentricities less than 0.06, so the circular orbit assumption seems valid. The slopes were determined using the the longest LUPI (8) VMAGT graphs in order to get the best estimate of the slope. The largest OPD (8) was also used. This gave the smallest slope of any LUPI/OPD combination for each satellite, and therefore the smallest angular rate of change. In the case of the Class 4-1-1 and 4-1-2 orbits, the LUPI 2, OPD 8 data was used. The data for this analysis is summarized in Table 6.2.

Table 6.2. First Order In-Track Error Analysis.

Class	Catalog Number	Semi-Major Axis	Eccentricity	In-Track Rate	Angular Rate
		km		km/day	rad/min
3 1 1	01996	7277	0.05379	12.5	1.2×10^{-6}
3 1 2	14443	7255	0.02245	1.2	1.2×10^{-7}
3 1 3	19643	7231	0.00670	2.5	2.4×10^{-7}
3 1 4	17429	7242	0.01095	2.5	2.4×10^{-7}
4 1 1	20335	6768	0.00175	75	1.0×10^{-5}
4 1 2	15584	6774	0.00234	75	1.0×10^{-5}

In all of these cases, by first-order approximation, the angular rate of change of the VMAGT error is at least 10 times larger than the mean motion variance limit ($\sqrt{10^{-15}} = 3.0 \times 10^{-8}$) used in the model. This was a strong indication that the mean motion variance needed to be limited in some way.

Once a mean motion variance limit, $\sigma^2_{n_{lim}}$, has been calculated and a determination made that the limit is indeed needed ($\sqrt{\sigma^2_n} < \sigma_{n_{lim}}$), there are two methods of implementing it within the DC program.

1. Diagonal adjustment. Apply the limit to only the mean motion variance term in the covariance matrix. This adjusted covariance matrix, P_a, is shown in Equation 6.1.

$$P_a = \begin{bmatrix} \sigma_{B^*}^2 & \sigma_{B^*}\sigma_i & \sigma_{B^*}\sigma_\Omega & \sigma_{B^*}\sigma_e & \sigma_{B^*}\sigma_\omega & \sigma_{B^*}\sigma_M & \sigma_{B^*}\sigma_n \\ \sigma_i\sigma_{B^*} & \sigma_i^2 & \sigma_i\sigma_\Omega & \sigma_i\sigma_e & \sigma_i\sigma_\omega & \sigma_i\sigma_M & \sigma_i\sigma_n \\ \sigma_\Omega\sigma_{B^*} & \sigma_\Omega\sigma_i & \sigma_\Omega^2 & \sigma_\Omega\sigma_e & \sigma_\Omega\sigma_\omega & \sigma_\Omega\sigma_M & \sigma_\Omega\sigma_n \\ \sigma_e\sigma_{B^*} & \sigma_e\sigma_i & \sigma_e\sigma_\Omega & \sigma_e^2 & \sigma_e\sigma_\omega & \sigma_e\sigma_M & \sigma_e\sigma_n \\ \sigma_\omega\sigma_{B^*} & \sigma_\omega\sigma_i & \sigma_\omega\sigma_\Omega & \sigma_\omega\sigma_e & \sigma_\omega^2 & \sigma_\omega\sigma_M & \sigma_\omega\sigma_n \\ \sigma_M\sigma_{B^*} & \sigma_M\sigma_i & \sigma_M\sigma_\Omega & \sigma_M\sigma_e & \sigma_M\sigma_\omega & \sigma_M^2 & \sigma_M\sigma_n \\ \sigma_n\sigma_{B^*} & \sigma_n\sigma_i & \sigma_n\sigma_\Omega & \sigma_n\sigma_e & \sigma_n\sigma_\omega & \sigma_n\sigma_M & \sigma_{n_{lim}}^2 \end{bmatrix} \tag{6.1}$$

2. Full adjustment. In addition to the diagonal $\sigma_{n_{lim}}^2$ term, apply the limit to the 12 other covariance elements (i.e., the elements in the bottom row and right column) of the covariance matrix impacted by σ_n. This would be implemented by first creating an adjustment matrix (I_a) as shown below:

$$I_a = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sigma_{n_{lim}}}{\sigma_n} \end{bmatrix} \tag{6.2}$$

The adjusted covariance matrix is formed by pre-multiplying and post-multiplying the covariance matrix by the adjustment matrix as shown in Equation 6.3.

$$P_a = I_a \, P \, I_a \tag{6.3}$$

The adjusted covariance matrix then takes the form shown in Equation 6.4.

$$P_a = \begin{bmatrix} \sigma_{B^*}^2 & \sigma_{B^*}\sigma_i & \sigma_{B^*}\sigma_\Omega & \sigma_{B^*}\sigma_e & \sigma_{B^*}\sigma_\omega & \sigma_{B^*}\sigma_M & \sigma_{B^*}\sigma_{n_{lim}} \\ \sigma_i\sigma_{B^*} & \sigma_i^2 & \sigma_i\sigma_\Omega & \sigma_i\sigma_e & \sigma_i\sigma_\omega & \sigma_i\sigma_M & \sigma_i\sigma_{n_{lim}} \\ \sigma_\Omega\sigma_{B^*} & \sigma_\Omega\sigma_i & \sigma_\Omega^2 & \sigma_\Omega\sigma_e & \sigma_\Omega\sigma_\omega & \sigma_\Omega\sigma_M & \sigma_\Omega\sigma_{n_{lim}} \\ \sigma_e\sigma_{B^*} & \sigma_e\sigma_i & \sigma_e\sigma_\Omega & \sigma_e^2 & \sigma_e\sigma_\omega & \sigma_e\sigma_M & \sigma_e\sigma_{n_{lim}} \\ \sigma_\omega\sigma_{B^*} & \sigma_\omega\sigma_i & \sigma_\omega\sigma_\Omega & \sigma_\omega\sigma_e & \sigma_\omega^2 & \sigma_\omega\sigma_M & \sigma_\omega\sigma_{n_{lim}} \\ \sigma_M\sigma_{B^*} & \sigma_M\sigma_i & \sigma_M\sigma_\Omega & \sigma_M\sigma_e & \sigma_M\sigma_\omega & \sigma_M^2 & \sigma_M\sigma_{n_{lim}} \\ \sigma_{n_{lim}}\sigma_{B^*} & \sigma_{n_{lim}}\sigma_i & \sigma_{n_{lim}}\sigma_\Omega & \sigma_{n_{lim}}\sigma_e & \sigma_{n_{lim}}\sigma_\omega & \sigma_{n_{lim}}\sigma_M & \sigma_{n_{lim}}^2 \end{bmatrix} \quad (6.4)$$

We attempted to determine the mean motion variance limit, $\sigma_{n_{lim}}^2$, required by the DC using our worst behaved class, 3-1-1 (Catalog Number 01996), as a test subject (reference Appendix H.7). A single random observation file was used to limit the processing time required by the DC. Additionally, for the purposed of this "tuning," we felt that difference between different random observation files would not change the order-of-magnitude limit we were searching for.

To tune the DC, we used the sum of all VMAGTs after Day 24 ($\sum_{i=24}^{N}$ VMAGT), as a performance measure. We selected day 24 because it was the default point at which steady-state was declared. Using both the diagonal and full adjustment methods discussed above, we ran the DC varying $\sigma_{n_{lim}}^2$ starting at 0.1. For each successive run we used a $\sigma_{n_{lim}}^2$ one order of magnitude lower. This process continued until the DC did not need to use the limit (i.e., $\sigma_{n_{lim}}^2 < \sigma_n^2$). The output from this process for each adjustment method was a listing of the incremented $\sigma_{n_{lim}}^2$ and the corresponding $\sum_{i=24}^{N}$ VMAGT. For each method, we chose as the limit the $\sigma_{n_{lim}}^2$ corresponding to the lowest $\sum_{i=24}^{N}$ VMAGT.

These limits were then applied and the DC model was run on all 16 LUPI/OPD combinations for this class. However, the VMAGT results for both adjustment methods showed little change. This led us to believe that the DC had to be tuned for all elements, not just mean motion.

Again, using the same class, 3-1-1, as a test subject, we determined variance and covariance limits for each element for both adjustment methods discussed above. Starting

with the element with the smallest variance, the variance limit for that element was calculated using the same method used to determine $\sigma^2_{n_{lim}}$ described above. That limit was then implemented (i.e., used in the DC if $\sqrt{\sigma^2} < \sigma_{lim}$) and the same process was used on the element with the next smallest variance. This continued until the variance limits for all elements was determined.

Using these 7 limits as an initial guess, the DC model was run, again using only one random observation file. The variance limit for each element was again varied while holding all other limits constant. The output from this process was a listing of the incremented $\sigma^2_{n_{lim}}$ and the corresponding $\sum_{i=24}^{N}$ VMAGT for each element. From this output, we determined the variance limit with the largest reduction in $\sum_{i=24}^{N}$ VMAGT between the limit used and an adjacent limit. The element variance limit was changed to the adjacent limit and the process repeated until the element variance limits resulting in the minimum $\sum_{i=24}^{N}$ VMAGT are determined. A basic assumption is that the local minimum determined by these variance limits is close to the global minimum. This process was performed for both adjustment methods.

Using these limits, the DC model was run on all 16 LUPI/OPD combinations for Class 3 1 1 (NORAD Catalog Number 01996). Both methods produced results much better than the baseline data seen in Appendix H.7. However, the full adjustment method did not provide results as accurate as the diagonal adjustment method. Reference Table 6.3 for the element covariances used in the diagonal adjustment and Appendix I.2 for the complete confidence interval analysis, ANOVA analysis and VMAGT graphs for the diagonal adjustment. Of special note is the statistical difference in results based on LUPI. Prior to tuning, there was no indicated difference in results. However, the tuning effort highlighted this difference.

To demonstrate the ability of this method to "tune" other orbital classes, the same methodology was applied to Class 1 3 2 (Catalog Number 14199). This class was chosen for three reasons. First, in the initial analysis, this class did not exhibit characteristics indicating the covariance had become too exact. Second, we wanted to see if "tuning" could improve previously "good" results. Third, this class is different from Class 3 1 1 in all three classification categories (i.e., mean motion, eccentricity, and inclination).

Table 6.3. Element Variance Limits Used for DC Tuning of Class 3-1-1.

Element Variance	Limit Imposed
$\sigma^2_{B^*}$	10^{-7} $(m^2/kg)^2$
σ^2_i	10^{-9} rad^2
σ^2_Ω	10^{-7} rad^2
σ^2_e	None
σ^2_ω	None
σ^2_M	None
σ^2_n	10^{-15} rad^2/min^2

Tuning for this class also improved overall VMAGT results. Reference Table 6.4 for the element variance limits used in the diagonal adjustment. Appendix I.1 contains the complete confidence interval analysis, ANOVA analysis and VMAGT graphs for the diagonal adjustment. ANOVA still indicates a difference in results based on LUPI, OPD, and the interaction of LUPI and OPD.

Table 6.4. Element Variance Limits Used for DC Tuning of Class 1-3-2.

Element Variance	Limit Imposed
$\sigma^2_{B^*}$	None
σ^2_i	10^{-9} rad^2
σ^2_Ω	10^{-8} rad^2
σ^2_e	10^{-10}
σ^2_ω	10^{-7} rad^2
σ^2_M	10^{-8} rad^2
σ^2_n	10^{-16} rad^2/min^2

6.3 *Differential Correction Model Performance*

As illustrated by most of the last-pass VMAGT graphs in Appendixes H and I, the differential corrector typically gave good performance. Some of the last-pass VMAGT graphs had results indicating that further information could be obtained from the data, such as the periodic results and occasional spikes in the data. However, in general, the randomly-distributed, steady-state nature of the data indicated good performance. Typically, the DC required between four and six iterations to converge for each batch of

observations, and only occasionally did not converge within the 50 iteration limit placed in the algorithm.

The actual run time of the program for each satellite investigated was quite long. Running on 25 MHz, 386 PC, a single 60-day run of the differential corrector, using one random observation file, an observation rate of two obs/day, and a single LUPI (between 8 and 30 corrections depending on LUPI) could be performed in approximately two minutes. However, to perform all 16 LUPI/OPD combinations using 10 random observation files (over 2500 corrections) took at least two and a half hours.

In addition, no singularity problems were encountered with the Keplerian elements at low eccentricities and low inclinations. An eccentricity and inclination of 0.0012 and 4.99° respectively were tested. These were both for Class 1-1-1.

VII. CONCLUSIONS AND RECOMMENDATIONS

7.1 Conclusions

Based on the 99 percent confidence level and ANOVA analysis of the simulation runs, we believe our model can be used to show the general differences in the accuracy of an orbital element set based on varying observation rates and correction intervals. In all of our test cases, we observed no results which indicate increasing LUPI or decreasing OPD would improve accuracy. Our results for mean motion Classes Two, Three, and Four showed either that accuracy improved as a result of increasing OPD or decreasing LUPI, or that no determination could be made due to large overlapping confidence intervals.

Mean motion Class One (deep space satellites) results were not that definitive. They showed that accuracy increased as a result of increasing OPD as expected, but in some cases, the LUPI effects may have been opposite what we expected. While the large overlapping confidence intervals make determinations difficult, the results of Class 1 3 2 after tuning (Reference Appendix I.1) could be interpreted as the VMAGTs for OPD/LUPI 8/8 being better than those for OPD/LUPI 8/2. This may be related to the LUPI dependent forgetting factor used in our model.

We believe the problem of the large overlapping confidence intervals is due to insufficient random observation runs for specific satellites. The need for more random observation runs showed up in the overlap of the 95 percent confidence interval of the 99 percent confidence level. This overlap made a determination of LUPI/OPD effects on results impossible for some satellites. We believe the addition of extra runs will narrow the confidence interval and allow better determination of results.

As a separate problem the accuracy of the reported VMAGT data indicates a need for the differential corrector (DC) to be "tuned" prior to use. On several of the output runs, the plotted first-pass VMAGT values demonstrated a significant positive "slope" characteristic. Apparently, this indicated the covariance had become too exact (very small). Therefore, the differential corrector would not allow the elements to be corrected. This problem was first identified early in test runs conducted to debug the DC code.

As indicated in Chapter IV, Section 4.4.1, we examined the impact of the covariance matrix on the correction process. To start the Bayes process, it was necessary to make an initial guess for the values in the matrix. The value of 10^{-7} was chosen based on an analysis of test data. As further test runs were conducted, we noticed the "slope" characteristic in all of the first-pass VMAGT plots. Preliminary analysis of the output indicated this may be the result of the DC "locking on" to a value of mean motion. Once locked on, the DC refused to correct mean motion and attempted to change other elements to make up the difference, thus creating a bad estimate.

As discussed in Chapter VI, we incorporated a constant mean motion variance-limiting factor into the code for the DC. The purpose of this factor was to prevent the DC from locking onto a specific value of mean motion and then refusing to correct it further. This procedure resulted in good first-pass corrections for several satellites, but not for all satellites. Further testing and analysis indicated this limiting factor is required on more than just the mean motion variance. Additionally, the element variance limits appear to be satellite/orbit dependent.

In our research, we manually tuned, or determined the variance limits necessary, for the two test cases analyzed. Based on the methodology described in Section 6.2, an automated process for performing this tuning for all orbit classes could be developed. With automated tuning, we believe our model could better determine the effects of LUPI and OPD on orbital element set accuracy.

7.2 *Follow-On Research Recommendations*

The most obvious follow-on research recommendation is the continuation of the project, as requested by our sponsors, the 1st Command and Control Squadron (1CACS), into the final development of a cost benefit assessment model. The results of this research are intended to be used as the basis for the development of a quantitative cost/benefit model to assess the effects of alterations to the SSN on orbit prediction accuracies. This model, as envisioned by personnel at the 1CACS, would be used as a decision making tool. The purpose of this tool would be to compare the gain or loss of orbit prediction accuracies with the costs or savings associated with the upgrade of a specific sensor site, the closing of

a specific sensor site, or the changing of sensor taskings. This decision-making tool would allow USSPACECOM to make better decisions (in terms of maintaining OE set accuracy) in regards to the SSN.

However, before attempting to incorporate our model into a larger application, an automated method for tuning the differential corrector is required. Coupled with this effort is the validation of the satellite classification method we used to select a representative sample from the satellite population. Based on the results of the classification method validation, a determination of whether the same variance limits can be applied to all satellites within a class, or must be determined independently for each satellite, can be made.

Other areas of recommended further research are:

- Re-coding of the model programs for computational efficiency as opposed to the easily interpretable format used.
- Examination of the effects of "tuning" the forgetting factor used in the differential correction process.
- Development of an interface process to ease the determination of the steady-state values for a representative set of satellites.
- Development of an interface process to allow execution of these models with various changes to SSN sensor characteristics (location and accuracies) to allow comparison to "baseline" run and, therefore, determine affects of alterations to SSN on element accuracy.
- Examination of the effects of "age de-weighting" the actual observations used in the sequential differential correction process.
- Creation of a task scheduling model. The results of this research could also be used as the basis for the development of a scheduling model to optimally schedule SSN sensors to gather only the observational data required to maintain the desired orbit prediction accuracies. This would ensure the SSC computers would process only the minimum required data to maintain the OE set accuracy.

Appendix A. *SGP LIBRARY DOCUMENTATION*

Documentation for
NORAD SGP4/SDP4 Units

Developed by
Dr TS Kelso

Version 2.50
1992 October 01

PURPOSE

The enclosed Pascal source code implements the NORAD SGP4/SDP4 orbital models for use with the standard two-line orbital element sets to determine earth-centered inertial (ECI) and topocentric coordinates of earth-orbiting objects. This code implements both the near-earth and deep-space portions of the NORAD SGP4 orbital model. These units are designed to make the development of programs based on the NORAD orbital models straightforward and standardized.

INTRODUCTION

In order to properly determine the position of any earth-orbiting object using the standard NORAD two-line element sets, it is necessary that the proper orbital model be used. Since the observations taken by NORAD for each earth-orbiting object are reduced to orbital elements using the SGP4 (Simplified General Perturbations) model, the SGP4 model *must* be used to get the most accurate determination possible of an object's position and velocity. The primary reason for this requirement is that each orbital model handles perturbations (due to atmospheric drag, solar and lunar gravitational effects, irregularities of the earth's gravitational field, etc.) in a different manner. The NORAD two-line element sets incorporate these perturbations using the SGP4 orbital model and that model is required to accurately reconstruct the magnitudes of these effects.

The SGP4 orbital model takes into account perturbations due to atmospheric drag (based on a static, non-rotating, spherically-symmetric atmosphere whose density can be described by a power law), fourth-order zonal geopotential harmonics (J2, J3, and J4), spin-orbit resonance effects for synchronous and semi-synchronous orbits, and solar and lunar gravitational effects to first order. The two portions of the SGP4 model are SGP4 (for objects in orbits with periods less than 225 minutes) and SDP4 (for objects in orbits greater than or equal to 225 minutes). The reason for breaking the model into two parts is that for low-earth orbits the effects of spin-orbit resonance and lunar and solar gravity are not significant. This result allows the development of an analytical model (SGP4) for determining an object's position and velocity, thereby reducing the computational burden. For deep-space orbits, a semi-analytical model (SDP4) is required.

Models which implement the older SGP model should be accurate for low-earth orbits but really won't be adequate for deep-space objects, particularly those in resonance with the geopotential.

COMPUTER IMPLEMENTATION

The enclosed Pascal source code was developed in Turbo Pascal Version 6.0 to fully implement the NORAD SGP4 orbital model. There are now twelve units provided:

SGP4SDP4	Full implementation of NORAD SGP4/SDP4 models
SGP_OBS	Observer-dependent routines for calculating topocentric information
MINMAX	Minimum/maximum functions
SGP_MATH	Various trigonometric and mathematical routines
SGP_TIME	Time-based routines for converting among time systems
SGP_INIT	Code and variables needed to initialize SGP4SDP4
SGP_INTF	Interface between SGP4SDP4 and SGP_CONV (and some special-purpose programs)
SGP_CONV	Routines for converting two-line data and SGP4 state vectors

SGP_IN	Routines to simplify input of data (with error checking)
SGP_OUT	Routines for outputting program results in standard formats
SOLAR	Routines for calculating the position of the sun and whether a satellite is in earth umbral eclipse
SUPPORT	General support routines for machine-dependent features

These units are structured to make development of software as simple as possible and to reduce the time needed for validating results. A certain amount of the development is also tailored toward the development of a similar set of units in C, which is now underway. A more complete description of these Pascal units is included in SGP4-PLB.ITF.

Of the units provided, the most complex is the implementation of the NORAD orbital models (SGP4SDP4). The development of this unit was done to follow as closely as possible the implementation contained in Spacetrack Report Number 3, "Models for Propagation of NORAD Element Sets," (a copy of the LaTeX documentation and complete FORTRAN source code is available on this system). No attempt has been made at this point to optimize the code to run faster. A future release will provide a more streamlined implementation. To ensure the validity of these units, no changes should be made to the units themselves which have been extensively tested and validated. LaTeX documentation to support this validation should be available on this system soon (look for SGP4-VAL.TEX).

A test program (SGP4TEST.PAS) is included to implement the sample cases included in the Spacetrack Report Number 3 documentation. It should provide a reasonably good idea of how to use these Turbo Pascal units. To determine an object's position and velocity, the object's two-line orbital element set is read into the array {sat_data} [Note: Variable names will be enclosed within braces to set them apart in this document]. A call is then made to the procedure Convert_Satellite_Data passing the satellite's index {satnumber}. In the call to this procedure, a determination is automatically made as to whether the object is in a near-earth or deep-space orbit; the result is returned in {ideep}. If {ideep} is 0, then the object is in a near-earth orbit; if {ideep} is 1, it is in a deep-space orbit. A call is then made to either SGP4 or SDP4 depending on the value of {ideep}.

The time passed in this call, {tsince}, represents the time since (before or after) the satellite epoch in the two-line element set. The four-dimensional vectors {pos} and {vel} are returned containing the x, y, and z ECI coordinates (referenced to the true equator and mean equinox of date) and vector magnitude of the object's position and velocity, respectively, at the specified time. Units are earth radii and earth radii per minute, respectively; appropriate conversions must be made to get the proper units (as demonstrated in the test program using Convert_Sat_State). The variable {iflag} is used internally for keeping track of initialization conditions for the deep-space portion of the model.

Running this test program should provide results quite close to those provided in the Spacetrack Report Number 3 documentation. The minor differences that do occur are due to two primary reasons. The first is the result of differences in the internal precision of the FORTRAN compiler used to generate the report and that of the Turbo Pascal compiler. The second is due to the choice to implement a consistent numerical precision in the Turbo Pascal code. Examination of the FORTRAN source code will reveal inconsistent use of single and double-precision variables and trigonometric functions; the Turbo Pascal code uses double-precision variables and functions throughout.

Because it is not very straightforward to perform calls to the SGP4 units based on the time since each satellite's element set epoch, a procedure is also provided which interfaces between a standard time system and the SGP4 and SDP4 calls. This procedure, SGP, requires only that the user pass the Julian Date of interest; the procedure Julian_Date_of_Epoch converts from a date in the format used in the two-line element set to a Julian Date. The SGP call also takes care of making the determination of whether the object is in a near-earth or deep-space orbit and calls the appropriate model. The SGP_TIME unit contains these procedures as well as routines to convert Julian Dates to more recognizable time formats.

The main advantage of using the Julian Date in SGP is that there is no problem calculating the time interval between the time of interest and the satellite element set epoch, regardless of whether this interval spans the beginning of a year. There is also a function to convert Julian Dates to calendar dates (Calendar_Date) of the form "1992 Jan 30." Epoch_Time transforms the Julian

Dates to the form used in the two-line element sets.

The test program SGP4TST2.PAS gives an example implementation of this approach. It is designed to produce the same results as SGP4TEST.PAS. The main difference is that the Julian Date is calculated to be the time of the epoch for each element set and then the offset (in minutes) is added. Be sure to note that times outside the units are now all in *days* not in minutes (as they are inside the units).

CONCLUSION

These Turbo Pascal units should make it easy to implement the official NORAD orbital models in developing any number of applications. It is now very straightforward to select input files (of satellite and observer data) and time conditions (with full error checking) and output data ranging from spacecraft ECI position and velocity to spacecraft ground tracks (latitude, longitude, and altitude) to look angles (azimuth, elevation, range, and range rate) to right ascension and declination. Each of these outputs uses the WGS '72 geoid (nonspherical) and takes into account atmospheric refraction, where appropriate. These results can be easily output as text or incorporated into advanced graphical applications.

For a full-blown example of how to implement these functions, look for the application TrakStar/SGP4, also available on this system. It allows output for up to 250 satellites (or element sets for a single satellite) in the form of spacecraft ECI position and velocity, spacecraft ground tracks (latitude, longitude, and altitude), look angles (azimuth, elevation, range, and range rate), and right ascension and declination. A separate data file is created for each object/element set -- a great tool for all sorts of analyses.

Future releases of these units will include routines to efficiently determine crossing phenomena (e.g., satellite rise/set, sun rise/set, and satellite entry into/exit from earth eclipse) and local optima (e.g., closest approach). And, of course, the existing units will be converted to C with an eye toward portability.

In a continuing effort to promote standardization of orbital calculations, I will endeavor to continue to improve this package and will gratefully accept user feedback or contributions to this effort.

- Dr TS Kelso
Internet: tkelso@afit.af.mil
anonymous ftp at archive.afit.af.mil
in the directory pub/space

SYSOP, Celestial BBS
513/427-0674 (modem)
Operating 24 hours/day
No parity, 8 data bits, 1 stop bit
300, 1200, 2400, 4800, and 9600 bps
v.32/32bis, v.42/42bis

2340 Raider Drive
Fairborn, OH, USA 45324-2001

Documentation for
NORAD SGP4/SDP4 Units

Interface Specifications

Developed by
Dr TS Kelso

Version 2.50
1992 October 01

INTRODUCTION

The following document is intended to provide complete documentation for the interface section of the units provided in this package. Each unit will be presented, in alphabetical order, and the INTERFACE section is reproduced, showing the exact nature of each call. Following this information will be a description of the intended use of each variable and procedure.

** UNIT MINMAX -- VERSION 1.02 **

```
Function IMin(arg1,arg2 : integer) : integer;
Function IMax(arg1,arg2 : integer) : integer;
Function RMin(arg1,arg2 : real) : real;
Function RMax(arg1,arg2 : real) : real;
Function DMin(arg1,arg2 : double) : double;
Function DMax(arg1,arg2 : double) : double;
```

These functions are used to provide integer, real, and double minimum and maximum calculations used in the libraries.

** UNIT SGP4SDP4 -- VERSION 1.50 **

```
  Uses Support,
       SGP_Init,
       SGP_Math,SGP_Time;

Procedure SGP(time : double;
       var pos,vel : vector);
Procedure SGP4(tsince : double;
             var iflag : integer;
           var pos,vel : vector);
Procedure SDP4(tsince : double;
             var iflag : integer;
           var pos,vel : vector);
```

This unit contains the Pascal implementation of the SGP4 and SDP4 orbital models. Two methods of interfacing are available. The first method is to access the individual routines exactly as described in Project Spacetrack Report Number 3 (Procedures SGP4 and SDP4). This method requires the user to calculate the time (in minutes) since each satellite element set epoch and determine the appropriate model to use. The variable (iflag) is used to keep track of initialization status and is set within these routines and within Convert_Satellite_Data (Unit SGP_CONV). This method is *NOT* recommended.

The second method is to access the orbital models via a call to SGP. Here the user is only required to pass a time (Julian Date) for the calculation and the satellite ECI position and velocity are returned. Determination of the appropriate orbital model is transparent to the user. The only requirement is that a call to Convert_Satellite_Data be made each time a new satellite is selected before calling SGP. Routines for calculating Julian Dates are included in Unit SGP_TIME.

** UNIT SGP_CONV -- VERSION 1.00 **

```
  Uses SGP_Math;

Procedure Convert_Satellite_Data(arg : integer);
Procedure Convert_Sat_State(var pos,vel : vector);
```

The first procedure is used to extract the data in a given two-line element set to the variables expected by UNIT SGP4SDP4. The data structures containing the two-line element sets is described in UNIT SGP_INIT. In addition, a determination as to whether the near-earth (SGP4) or deep-space (SDP4) model should be used is made within this unit. These variables are passed between SGP4SDP4 and SGP_CONV using UNIT SGP_INTF. This unit (SGP_INTF) should NOT be included in the main program unless absolutely necessary to avoid unintentional changes to these critical variables. Two additional variables, (catnr) and (elset), are provided to identify the satellite element set. Finally, the variable (epoch) is globally available to determine the epoch of the element set.

Convert_Sat_State is used to convert the native units provided by SGP4/SDP4 (position in earth radii, velocity in earth radii/minute) to standard metric

units (kilometers and kilometers/sec).

** UNIT SGP_IN -- VERSION 2.00 ***

```
(** This unit contains machine-specific code **)

  Uses SGP_Math,SGP_Init,Support;

const
  data_type : byte = 3;

var
  fsat,fobs : text;

Procedure Select_Time(message : string;
                     xpos,ypos : byte;
                   var default : time_set;
                     precision : byte);
Procedure Select_Time_Interval(message : string;
                              xpos,ypos : byte;
                            var default : time_set;
                              precision : byte);
Function Checksum_Good(line : line_data) : boolean;
Function Good_Elements(lines : two_line) : boolean;
Procedure Input_Satellite(index : word);
Function Input_Satellite_Data(fn : string) : word;
Procedure Select_Satellites(title : string;
                            x,y,w,h : byte;
                             number : word);
Procedure Input_Observer(var geodetic : vector);
```

This unit is one of two units which contain machine-specific code; this limitation will probably be removed in the future, moving all machine-specific code to UNIT SUPPORT.

The constant {data_type} is preset to 3 and is used to specify the format of the two-line orbital data file. A value of 2 indicates that the satellite data files contain *only* two-line data; a value of 3 indicates that each two-line element set is preceded by a 22-character satellite name. These data are read into data structures as described in UNIT SGP_INIT. The value of {data_type} can be changed in the main program and, if necessary, should be done within the program initialization block.

The variables {fsat} and {fobs} are predefined text files to be assigned to the satellite and observer files, respectively. Routines used in this unit expect to use these file handles.

The first two procedures, Select_Time and Select_Time_Interval, are provided to make it easy to input start/stop times and time intervals for performing calculations. Each procedure will put a window on the screen with the title specified by {message} at the location ({xpos},{ypos}). A default time set is passed to initialize the time; the structure of this variable is set forth in UNIT SUPPORT. The variable {default} may be initialized manually or by a call to Get_Current_Time (for Select_Time) or Zero_Time (for Select_Time_Interval); each of these procedures is also in UNIT SUPPORT. The variable {precision} is used to indicate the precision of the data to be selected. Precision ranges from years down to hundredths of a second. A precision of 7 indicates that all time units are to be input; a precision of 5 would omit inputting seconds and hundredths of seconds. All values beyond the specified precision are set to zero, regardless of any other user action.

The next two functions are used to determine whether a given two-line element set is "good" -- at least in the sense that it passes the modulo-10 checksum on each line and that all the numbers appear to be in the proper fields. A simple call passing the two-line element set {lines} to Good_Elements returns TRUE if the data passes these tests. The function Checksum_Good can be used to test the modulo-10 checksum for a single line of a two-line element set. These functions are not used explicitly during element set input as it is assumed that the user has already done so (all data that I post has already been subjected to these tests). In fact, the program PASSUPDT, which is available on this system for updating files of two-line element sets from a master file, does this checking for you. However, if you are unsure of the quality of your data, it is STRONGLY RECOMMENDED that you develop your own preprocessor using these functions to test your data.

The procedure Input_Satellite is used to read in individual two-line element sets. The index indicates its placement within the data array. This call assumes that {fsat} has already been ASSIGNed and initialized. Typically, this call is made from the function Input_Satellite_Data which initializes {fsat} and reads an entire file into memory from the file {fn}; the output of this function is the number of satellites in {fn}. However, due to memory limitations, some files will be too large to read into memory. In these cases, Input_Satellite should be used to sequentially read and process individual element sets, each element set could be read into {index}=1 and

processed before proceeding.

In cases where all the orbital data can be read into memory and {data_type}=3, the procedure Select_Satellites can be used to tag satellites for calculations later in the program using the array {selected}. Initially, no satellites are selected. A call to this procedure will place a window on the screen with the upper-left corner at ({x},{y}) and having a width {w} and maximum height {h} (the height ranges from 1 to the minimum of {h} and the number of satellites). A {title} is also put on the window. The final parameter in this call is the number of satellites available. Selection is achieved by moving with the up/down cursor keys to the appropriate satellite and toggling with the space bar (an asterisk marks an item as being selected); data scrolls in the window as necessary. Toggling advances to the next item in the list, making it easier to quickly mark items. The 'A' key toggles All items (negates their current status). Once the desired items are selected, the [ENTER] key completes the process. Within the main program, items can be tested for selection with a statement such as: if selected[index] then ...

The final procedure for this unit is Input_Observer and it works somewhat like Input_Satellite except that the data is read into a {geodetic} four-vector where {geodetic}[1] is North latitude, {geodetic}[2] is East longitude (that means that West longitudes are negative), and {geodetic}[3] is altitude above mean sea level (AMSL). Input is expected from the file {fobs} in units of degrees and meters but is immediately converted to radians and kilometers for use within these units. Each observer entry in {fobs} consists of a single line beginning with a 25-character site name (passed as {obs_name}); it is expected that this name has a three-character short name (or number), two spaces, followed by a long name. The short name is used within UNIT SGP_OUT for topocentric output. The remaining three numbers on this line are free-field format, beginning on or after character 26.

** UNIT SGP_INIT -- VERSION 1.10 ***

```
const
  max_sats = 250;

type
  line_data = string[69];
  two_line  = array [1..2] of line_data;

var
  visible               : boolean;
  epoch                 : double;
  catnr,elset           : string;
  obs_name              : string[25];
  selected              : array [1..max_sats] of boolean;
  sat_name              : array [1..max_sats] of string[22];
  sat_data              : array [1..max_sats] of two_line;
  data_drive,data_dir,
  work_drive,work_dir   : string;

Procedure Program_Initialize(program_name : string);
```

This unit is used to initialize most of the data structures specific to the SGP4 family of units. The constant {max_sats} sets the limit (imposed by system memory constraints) on the maximum number of satellites available. This constant affects the storage allocation for two-line element sets in the array {sat_data}, satellite names in the array {sat_name}, and the array of selected satellites in {selected}. Note that type {two_line} is a two-element string of 69-character lines.

The first variable provided is a boolean variable {visible} which is set when making calls to determine topocentric position; if the satellite is visible to the observer, {visible} is set true, otherwise it is set false. Checks of this variable make sense only in this context. Eventually, this variable will also be used in determining other visibility conditions.

The next three variables pertain to the current two-line element set (the last one processed through Convert_Satellite_Data). The variables {catnr} and {elset} are read from the appropriate field of Line 1 and serve to identify the data. The variable {epoch} marks the epoch time (in two-line format) of that element set.

The final four variables are determined in the procedure Program_Initialize. They are used to specify the location of data files ({fsat} and {fobs} files) and output files. Having separate locations for these two groups makes development and testing easier (in my opinion) and allows for standard locations to facilitate integration with other programs (it doesn't make sense to have various versions of two-line element set files scattered all over your hard disk). These values are specified in the configuration file {program_name}.CFG (if it exists; if not, everything defaults to the working disk and directory). For example, in the program TRAKSTAR, there would be a

file TRAKSTAR.CFG which might look like:

```
d:
\data\
e:
\work\
```

If this program were located in C:\SGP4 and the data was in C:\SGP4\DATA and work files were to go to C:\SGP4\WORK, the configuration file might look like:

```
c:
\sgp4\data\
c:
\sgp4\work\
```

Note that each directory ends with a trailing \ (and should have no blanks). If you wish to run from the default drive (useful when using removable cartridges and moving between systems), you might change the above to:

```
\sgp4\data\

\sgp4\work\
```

To specify directories which are below the working directory, you could also use:

```
data\

work\
```

The call to Program_Initialize will read {program_name}.CFG to determine this information (if it exists) and will also read {program_name}.HDR (if it exists) to place one page of information on the screen to identify the program. It is strongly recommended that these parameters be used in your programs to facilitate traceability and portability.

A call to Program_End is strongly recommended to reset the terminal after program execution. Currently, this procedure positions the cursor to the bottom of the screen and makes sure it is on.

** UNIT SGP_INTF -- VERSION 1.02 ***

```
const
  ae       = 1;
  tothrd   = 2/3;
  xkmper   = 6378.135;        {Earth equatorial radius - kilometers (WGS '72)}
  f        = 1/298.26;        {Earth flattening (WGS '72)}
  ge       = 398600.8;        {Earth gravitational constant (WGS '72)}
  J2       = 1.0826158E-3;    {J2 harmonic (WGS '72)}
  J3       = -2.53881E-6;     {J3 harmonic (WGS '72)}
  J4       = -1.65597E-6;     {J4 harmonic (WGS '72)}
  ck2      = J2/2;
  ck4      = -3*J4/8;
  xj3      = J3;
  qo       = ae + 120/xkmper;
  s        = ae + 78/xkmper;
  e6a      = 1E-6;
  dpinit   = 1;               {Deep-space initialization code}
  dpsec    = 2;               {Deep-space secular code}
  dpper    = 3;               {Deep-space periodic code}

var
  iflag,ideep                 : integer;
  xmo,xnodeo,omegao,eo,xincl,
  xno,xndt2o,xndd6o,bstar,
  julian_epoch,xke            : double;
```

This unit defines the constants and variables used internal to SGP4SDP4. Note that this version still uses WGS '72 values (while the impact of switching to WGS '84 is assessed). It is STRONGLY recommended that this unit NOT be included in the main program. All determinations using these constants (such as position of an observer on the earth's surface in the ECI system) should be performed through the appropriate call. Also, note that {iflag} and {ideep} are unavailable to the main program under this recommendation, forcing the use of the procedure SGP to access the NORAD orbital models

** UNIT SGP_MATH -- VERSION 1.30 ***

```
type
  vector = array [1..4] of double;

const
  twopi = 2 * pi;
  zero  : vector = (0,0,0,0);

Function Sign(arg : double) : shortint;
Function Cube(arg : double) : double;
```

```
Function Power(arg,pwr : double) : double;
Function Radians(arg : double) : double;
Function Degrees(arg : double) : double;
Function Tan(arg : double) : double;
Function ArcSin(arg : double) : double;
Function ArcCos(arg : double) : double;
Function Modulus(arg1,arg2 : double) : double;
Function Fmod2p(arg : double) : double;
Function AcTan(sinx,cosx : double) : double;
Function Dot(v1,v2 : vector) : double;
Procedure Magnitude(var v : vector);
Procedure Cross(v1,v2 : vector; var v3 : vector);
```

This unit first defines the the type {vector} is as a four-element array (or four-vector) consisting (typically) of x, y, and z position and a vector magnitude. These vectors are also used for storing other types of information (as will be seen in UNIT SGP_OBS). Next, the constant {twopi} is defined as appropriate for Turbo Pascal Version 6.0. Note that earlier versions of Turbo Pascal will NOT allow constant definitions of this form. The definition of a zero vector has been added for this release.

The remaining routines are defined below.

Function Sign(arg : double) : shortint;
    Output is the sign (-1, 0, +1) of {arg}.
Function Cube(arg : double) : double;
    Output is {arg} to the third power.
Function Power(arg,pwr : double) : double;
    Output is {arg} to the {pwr} power. This is a more general-purpose function, but is restricted to positive values of {arg}. An error message is reported if {arg} violates this restriction.
Function Radians(arg : double) : double;
    Output is an angle in radians where {arg} was input in degrees.
Function Degrees(arg : double) : double;
    Output is an angle is degrees where {arg} was input in radians.
Function Tan(arg : double) : double;
    Output is the tangent of {arg}.
Function ArcSin(arg : double) : double;
    Output is the arcsine of {arg} in radians. Values range between -pi and +pi.
Function ArcCos(arg : double) : double;
    Output is the arccosine of {arg} in radians. Values range between 0 and 2*pi.
Function Modulus(arg1,arg2 : double) : double;
    Output is the remainder after {arg1} is divided by {arg2}. This routine is useful for keeping angles between 0 and 2*pi (or 0 and 360 degrees).
Function Fmod2p(arg : double) : double;
    This function is a specific implementation of Modulus where {arg2} equals 2*pi. Used explicitly within SGP4SDP4.
Function AcTan(sinx,cosx : double) : double;
    Output is the arctangent of {sinx}/{cosx} in radians. Used explicitly within SGP4SDP4. The advantage of this function over ArcTan is that it returns the correct quadrant of the angle.
Function Dot(v1,v2 : vector) : double;
    Output is the vector dot product of {v1} and {v2}.
Procedure Magnitude(var v : vector);
    This procedure calculates the magnitude of vector {v} using components 1, 2, and 3, storing the result in component 4.
Procedure Cross(v1,v2 : vector; var v3 : vector);
    Returns vector {v3} which is the vector cross product of {v1} and {v2}.

```
** UNIT SGP_OBS -- VERSION 1.40 ************************************************

  Uses SGP_Math;

Procedure Calculate_User_PosVel(var geodetic  : vector;
                                    time      : double;
                            var obs_pos,obs_vel : vector);
Procedure Calculate_LatLonAlt(pos : vector;
                              time : double;
                         var geodetic : vector);
Procedure Calculate_Obs(pos,vel,geodetic : vector;
                                 time    : double;
                         var obs_set     : vector);
Procedure Calculate_RADec(pos,vel,geodetic : vector;
                                  time     : double;
                           var obs_set     : vector);
```

These procedures are used to provide observer-specific coordinate

transformations of the output of the NORAD orbital models.

Procedure Calculate_User_PosVel passes the user's geodetic position and the time of interest and returns the ECI position and velocity of the observer. The format of {geodetic} is as explained in UNIT SGP_IN. The velocity calculation assumes the geodetic position is stationary relative to the earth's surface. This routine is made available to the user, although the exact nature of its application is uncertain. This procedure is used, however, as the basis for Calculate_Obs and Calculate_RADec, to be explained later.

Procedure Calculate_LatLonAlt will calculate the {geodetic} position of an object given its ECI position {pos} and {time}. It is intended to be used to determine the ground track of a satellite. The calculations assume the earth to be an oblate spheroid as defined in WGS '72.

The procedures Calculate_Obs and Calculate_RADec calculate the *topocentric* coordinates of the object with ECI position, {pos}, and velocity, {vel}, from location {geodetic} at {time}. The {obs_set} returned for Calculate_Obs consists of azimuth, elevation, range, and range rate (in that order) with units of radians, radians, kilometers, and kilometers/second, respectively. The WGS '72 geoid is used and the effect of atmospheric refraction (under standard temperature and pressure) is incorporated into the elevation calculation; the effect on range and range rate has not yet been quantified.

The {obs_set} for Calculate_RADec consists of right ascension and declination (in that order) in radians. Again, calculations are based on *topocentric* position using the WGS '72 geoid and incorporating atmospheric refraction.

```
** UNIT SGP_OUT -- VERSION 1.50 **********************************************

  Uses SGP_Math;

const
  day_date  : boolean = true;
  full_time : boolean = true;
  N_E_W_S   : boolean = false;
  D_M_S     : boolean = false;
  time_res  : byte = 2;
  angle_res : byte = 4;
  dist_res  : byte = 3;

var
  fout : text;

Procedure Output_Time(time : double);
Procedure Output_ECI(time : double;
                     pos,vel : vector);
Procedure Output_Angle(angle : double;
                       width,dec : byte;
                         degrees : boolean);
Procedure Output_LatLonAlt(time : double;
                           geodetic : vector);
Procedure Output_Obs(time : double;
                     obs : vector);
Procedure Output_RADec(time : double;
                       obs : vector);
```

This unit begins with a set of constants used as formatting parameters. The intention here is to set these values for consistency in the output and not clutter output calls with formatting information. While the default values are set above, it is easy to change these values either globally or locally. For example, the user may decide to change these values in the program initialization step. Or, these values could be set prior to making calls to routines within UNIT SGP_OUT.

The constant {day_date} allows for selection of output time as day/date (if true) or as a Julian Date (easier to read into other programs).

The constant {full_time} allows the selection of times with colons between the hours and minutes and between the minutes and seconds (i.e., HH MM SS), if set true.

The constant {N_E_W_S} is used with outputs of latitude and longitude to specify an output with North/South or East/West (if true) rather than plus or minus values.

The constant {D_M_S} allows for angular output in degrees, minutes, and seconds (if true) rather than decimal degrees.

The final three constants are used to set the precision of output for time, angle, and distance variables, respectively. The default for {time_res} of 2 indicates output to hundredths of a second. The default for {angle_res} of 4 indicates output to four decimal places (if D_M_S = false) or arcseconds (if D_M_S = true), outputs in degrees, minutes, and seconds are rounded to tens of

arcminutes, arcminutes, tens of arcseconds, and seconds for values of 1, 2, 3, and 4, respectively. The default for {dist_res} of 3 indicates a precision to three decimal places in distances (i.e., meters) and six decimal places in velocities (i.e., millimeters/second).

The variable {fout} is a pre-defined variable to specify the output file.

The procedure Output_Time is used to output either the calendar date and time of day or Julian Date of {time} to {fout}.

The procedure Output_ECI outputs ECI position {pos} (in kilometers) and velocity {vel} (in kilometers/second) vectors along with the time.

The procedure Output_Angle outputs an angle {angle} in a field of width {width} with {dec} decimal places. The variable {degrees} is used when {D_M_S} is true to specify whether the angular output is degrees or hours.

The procedure Output_LatLonAlt output the time, latitude (in degrees), longitude (in degrees), and altitude (in kilometers) of an object.

The procedure Output_Obs outputs the time, azimuth (in degrees), elevation (in degrees), range (in kilometers), and range rate (in kilometers/second with negative values approaching the observer).

The procedure Output_RADec outputs the time, right ascension (in hours), and declination (in degrees).

Output_ECI and Output_LatLonAlt now write ECL at the end of each line if the satellite is in earth umbral eclipse.

Output_Obs and Output_RADec now output a blank line at the completion of a pass.

** UNIT SGP_TIME -- VERSION 1.50 **

```
  Uses SGP_Math;

type
  clock_time = string[12];
  date       = string[11];

const
  xmnpda   =  1440.0;         {Minutes per day}
  secday   = 86400.0;         {Seconds per day}
  omega_E  = 1.00273790934;   {Earth rotations per sidereal day (non-constant)}
  omega_ER = omega_E*twopi;   {Earth rotation, radians per sidereal day}

var
  ds50 : double;

Function Julian_Date_of_Year(year : double) : double;
Function Julian_Date_of_Epoch(epoch : double) : double;
Function Epoch_Time(jd : double) : double;
Function DOY(yr,mo,dy : word) : word;
Function Fraction_of_Day(hr,mi,se,hu : word) : double;
Function Calendar_Date(jd : double) : date;
Function Time_of_Day(jd : double;
                     full : boolean;
                      res : byte) : clock_time;
Function ThetaG(epoch : double) : double;
Function ThetaG_JD(jd : double) : double;
Function Delta_ET(year : double) : double;
```

This unit contains all the routines to perform time conversions.

The function Julian_Date_of_Year calculates the Julian Date of Day 0.0 of {year}. This function is used to calculate the Julian Date of any date by using Julian_Date_of_Year, DOY, and Fraction_of_Day.

The function Julian_Date_of_Epoch returns the Julian Date of an epoch specified in the format used in the NORAD two-line element sets

The function DOY calculates the day of the year for the specified date. The calculation uses the rules for the Gregorian calendar and is valid from the inception of that calendar system

Fraction_of_Day calculates the fraction of a day passed at the specified input time

The function Calendar_Date converts a Julian Date to a string of the form "yyyy Mon dd" It is typically used as the major output time format

Time_of_Day takes a Julian Date and calculates the clock time portion of that date The variable {full} is set true if it is desired to place colons between hours and minutes and minutes and seconds The variable {res} is used to determine the number of decimal places after the seconds in the output. zero gives a resolution of seconds, one gives a resolution of tenths of

seconds, etc. The variable {res} can take on values between 0 and 3.

The function ThetaG calculates the Greenwich Mean Sidereal Time for an epoch specified in the format used in the NORAD two-line element sets. The function ThetaG_JD provides the same calculation except that it is based on an input in the form of a Julian Date.

The function Delta_ET has been added to allow calculations on the position of the sun. It provides the difference between UT (approximately the same as UTC) and ET (now referred to as TDT). This function is based on a least squares fit of data from 1950 to 1991 and will need to be updated periodically.

** UNIT SOLAR -- VERSION 1.20 **

```
  Uses SGP_Math;

const
  eclipsed : boolean = false;
  show_vis : boolean = false;

var
  civil,
  nautical,
  astronomical : double;  {Twilight elevations}
  solar_pos    : vector;

Procedure Calculate_Solar_Position(time : double;
                        var solar_vector : vector);
Function Depth_of_Eclipse(time : double;
                          r1 : vector) : double;
```

The variable {eclipsed} is provided for use with the Function Depth_of_Eclipse to reflect whether a satellite is in earth umbral eclipse The constant {show_vis} is used to indicate whether information should be output for all passes or only visible passes (i.e., satellite above the observer's horizon, not in earth umbral eclipse, and the sun below the appropriate threshold for skies to be dark at the observer's location).

The variables {civil}, {nautical}, and {astronomical} are initialized to the thresholds for solar elevation defining the corresponding twilights. These are defined to occur at 6, 12, and 18 degrees below the horizon, respectively.

The variable {solar_pos} is provided to track the position of the sun as calculated in Depth_of_Eclipse, thus avoiding an additional call to Calculate_Solar_Position.

The procedure Calculate_Solar_Position calculates the position of the sun in the ECI coordinate system at the designated time (UTC). The procedure makes an adjustment for the difference between UT and TDT by calling Delta_ET in Unit SGP_TIME. The variable {solar_vector} returns the ECI position in kilometers and calculates the distance to the sun.

The function Depth_of_Eclipse calculates the distance an object at position {r1} at {time} is into the cone enclosing both the earth and the sun. On the side of the earth away from the sun, this cone defines the earth's umbral shadow. This distance is determined by first finding the perpendicular distance of the object from the line going through the centers of both the earth and the sun and then calculating the distance of the edge of the cone from this same line along the perpendicular. The depth into eclipse is defined as the difference of these two distances. Positive values reflect distances outside the eclipse zone. Because the object can be within the cone on either side of the earth, the variable {eclipsed} is set true if the object is within the cone on the side of the earth opposite of the sun

** UNIT SUPPORT -- VERSION 1.31 **

```
const  {IBM PC screen codes}
  BS     = ^H,                   {Backspace}
  CR     = ^M,                   {Carriage Return}
  CRLF   = ^M^J,                 {Carriage Return/Line Feed}
  BELL   = ^G,                   {Terminal Bell}
  ESC    = ^[,                   {Escape}
  DEL    = #$7F,                 {Delete}
  Up     = #72,                  {Up Cursor}
  Dn     = #80,                  {Down Cursor}
  Rt     = #77,                  {Right Cursor}
  Lt     = #75,                  {Left Cursor}
  Home   = #71,                  {Home Key}
  Endd   = #79,                  {End Key}
  PgUp   = #73,                  {Page Up}
  PgDn   = #81,                  {Page Down}
  C_Lt   = #115,                 {Control-Left Cursor}
  C_Rt   = #116,                 {Control-Right Cursor}
  C_PgUp = #132,                 {Control-Page Up}
```

```
  C_PgDn   =  #118;                    {Control-Page Down}
  UpDown   =  #24#25;                  {Up/Down Arrows}
  Cursors  =  #24#25#26#27;            {Up/Down/Left/Right Arrows}
  SFrame : string = '';      {Single-Line Frame Characters}
  DFrame : string = '';      {Double-Line Frame Characters}
  MFrame : string = '';           {Mixed-Line Frame Characters}

type
  options  = array [0..10] of string;
  time_set = record
    yr,mo,dy,hr,mi,se,hu : word;
    end; {record}

Procedure Cursor_On;
Procedure Cursor_Off;
Procedure Save_Cursor;
Procedure Restore_Cursor;
Procedure ReverseVideo;
Procedure NormalVideo;
Procedure BoldVideo;
Procedure FrameWindow(x,y,w,h,color : byte; title : string);
Procedure MakeWindow(x,y,w,h,color : byte; title : string);
Procedure ClearWindow(x,y,w,h : byte);
Procedure Show_Status_Line(title : string);
Procedure Show_Instructions(title : string);
Procedure Clear_Status_Line;
Procedure Report_Error(x,y : byte; title : string);
Procedure Beep;
Procedure Buzz;
Procedure Mark_Time;
Procedure Zero_Time(var time : time_set);
Procedure Get_Current_Time(var time : time_set);
Function Yes : boolean;
Function TwoDigit(arg : integer) : string;
Function ThreeDigit(arg : integer) : string;
Procedure Convert_Blanks(var field : string);
Function Integer_Value(buffer : string;
                 start,length : integer) : integer;
Function Real_Value(buffer : string;
              start,length : integer) : double;
Function File_Exists(filename : string) : boolean;
Function Select_File(title,pattern,default : string; x,y,w,h : byte) : string;
Function Select_Option(menu : options; number,x,y,w,h : byte) : byte;
```

This unit is considerably different from the other units in this library and is intended to be a general support library for a specific hardware implementation, in this case the IBM PC compatible family of microcomputers running Microsoft DOS. Eventually, all machine-specific code will be put in this unit to enhance portability, meaning that only these routines will have to be changed to port it to another machine. I have decided not to delay the release of the package to achieve this goal since there is no widely available standard implementation of Pascal to necessitate such changes. However, when these units are converted to C, all machine-specific routines will be limited to this unit.

The unit begins with a set of constants which define the specifics of various keys and characters on IBM PCs; their definitions are commented above.

The two types defined are an array of {options} which are used with the function Select_Option and {time_set} which is a record containing the year, month, day, hour, minute, second, and hundredth of seconds.

The remaining routines are described below.

Procedure Cursor_On.
    Turns on a block cursor on the screen at the current cursor location.
Procedure Cursor_Off.
    Hides the cursor from view.
Procedure Save_Cursor.
    Saves the current position of the cursor.
Procedure Restore_Cursor.
    Restores the cursor location to the last saved location.
Procedure ReverseVideo.
    Changes the text attributes to reverse video (black on white).
Procedure NormalVideo.
    Changes the text attributes to normal video (white on black).
Procedure BoldVideo.
    Changes the text attributes to bold (yellow on black).
Procedure FrameWindow(x,y,w,h,color : byte; title : string).
    Puts a frame around a window (and is usually called by MakeWindow).
Procedure MakeWindow(x,y,w,h,color : byte; title : string).
    Makes a window, located at ({x},{y}) with width {w} and height {h} in color {color}. A {title} is put on the window.

Procedure ClearWindow(x,y,w,h : byte);
ClearWindow simply clears a window from the display. The code does not, at this time, allow restoration of underlying screen text.

Procedure Show_Status_Line(title : string);
Shows status on the left side of the last line of the display. Usually used in support of providing brief instructions during input.

Procedure Show_Instructions(title : string);
Similar to Show_Status_Line except providing right-justified output on the last line of the display.

Procedure Clear_Status_Line;
Clears the last line of the display.

Procedure Report_Error(x,y : byte; title : string);
Reports the error specified by {title} in BoldVideo at the position ({x},{y}) and then exits the program.

Procedure Beep;
Sounds a high-pitched tone to draw attention.

Procedure Buzz;
Sounds a lower-pitched tone to indicate an error condition

Procedure Mark_Time;
A rotating cursor to serve as a visual indication that the machine is still operating during time-intensive operations.

Procedure Zero_Time(var time : time_set);
Places zeros in all element of {time}. Useful in initializing in preparation for a call to Select_Time_Interval.

Procedure Get_Current_Time(var time : time_set);
Loads the current system time into variable {time}. Useful for initializing Select_Time.

Function Yes : boolean;
A function which waits for the [illegible] of a Y or N in response to a question. The words Yes or No [illegible] in response to the appropriate input and Yes is set true if a [illegible]ve response is received. For example,

```
Write('Are you ready? ');
if Yes then Do_Ready else Do_Not_Ready;
```

Function TwoDigit(arg : integer) : string;
Converts an integer to a two-digit string with leading zeros.

Function ThreeDigit(arg : integer) : string;
Converts an integer to a three-digit string with leading zeros.

Procedure Convert_Blanks(var field : string);
Used with the next two functions to convert leading spaces to zeros to facilitate text conversion to integers or reals.

Function Integer_Value(buffer : string;
start,length : integer) : integer;
Takes the segment of {buffer} beginning at {start} and having length {length} and converts it to an integer number.

Function Real_Value(buffer : string;
start,length : integer) : double;
Takes the segment of {buffer} beginning at {start} and having length {length} and converts it to a double precision real number.

Function File_Exists(filename : string) : boolean;
Checks to see if {filename} exists.

Function Select_File(title,pattern,default : string; x,y,w,h : byte) : string;
Allows for easy selection of input files. Select_File places a window on the screen with upper-left corner at location ({x},{y}) and having width {w} and maximum height of {h}. The window is labeled with {title} and {pattern} specifies the pattern of files to look for; {default} is the default pattern (this filename is highlighted, if present). Select_File returns the filename selected.

Function Select_Option(menu : options; number,x,y,w,h : byte) : byte;
Allows for the easy selection of an option from a menu {menu}. The values of {x}, {y}, {w}, and {h} are the same as for Select_File; {number} indicates the total number of options available. Select_Option returns the option number selected.

Appendix B. *SGP LIBRARY SOURCE LISTINGS*

B.1 SGP4SDP4 Unit Source Code Listing

```
Unit SGP4SDP4;
{           Author:  Dr TS Kelso }
{ Original Version:  1991 Oct 30}
{ Current Revision:  1992 Sep 03}
{          Version:  1.50 }
{        Copyright:  1991-1992, All Rights Reserved }
{$N+}

INTERFACE
  Uses Support,
       SGP_Init,
       SGP_Math,SGP_Time;

Procedure SGP(time : double;
      var pos,vel : vector);
Procedure SGP4(tsince : double;
            var iflag : integer;
          var pos,vel : vector);
Procedure SDP4(tsince : double;
            var iflag : integer;
          var pos,vel : vector);

IMPLEMENTATION
  Uses SGP_Intf;

var
{dpinit}
  eqsq,siniq,cosiq,rteqsq,ao,cosq2,sinomo,cosomo,
  bsq,xlldot,omgdt,xnodot,xnodp                       : double;
{dpsec/dpper}
  xll,omgasm,xnodes,_em,xinc,xn,t                     : double;
  qoms2t                                              : double;

Procedure Define_Derived_Constants;
  begin
  xke    := Sqrt(3600*ge/Cube(xkmper));  {Sqrt(ge) ER^3/min^2}
  qoms2t := Sqr(Sqr(qo-s));              {(qo-s)^4 ER^4}
  end; {Define_Derived_Constants}

Procedure SGP4(tsince : double;
            var iflag : integer;
          var pos,vel : vector);
  label
    9,10,90,100,110,130,140;
  const
    a1     : double = 0;  a3ovk2 : double = 0;  ao     : double = 0;
    aodp   : double = 0;  aycof  : double = 0;  betac  : double = 0;
    betao2 : double = 0;  c1     : double = 0;  c1sq   : double = 0;
    c2     : double = 0;  c3     : double = 0;  c4     : double = 0;
    c5     : double = 0;  coef   : double = 0;  coef1  : double = 0;
    cosio  : double = 0;  d2     : double = 0;  d3     : double = 0;
    d4     : double = 0;  del1   : double = 0;  delmo  : double = 0;
    delo   : double = 0;  eeta   : double = 0;  eosq   : double = 0;
    eta    : double = 0;  etasq  : double = 0;  isimp  : integer = 0;
    omgcof : double = 0;  omgdot : double = 0;  perige : double = 0;
    pinvsq : double = 0;  psisq  : double = 0;  qoms24 : double = 0;
    s4     : double = 0;  sinio  : double = 0;  sinmo  : double = 0;
    t2cof  : double = 0;  t3cof  : double = 0;  t4cof  : double = 0;
    t5cof  : double = 0;  temp   : double = 0;  temp1  : double = 0;
    temp2  : double = 0;  temp3  : double = 0;  theta2 : double = 0;
    theta4 : double = 0;  tsi    : double = 0;  x1m5th : double = 0;
    x1mth2 : double = 0;  x3thm1 : double = 0;  x7thm1 : double = 0;
    xhdot1 : double = 0;  xlcof  : double = 0;  xmcof  : double = 0;
    xmdot  : double = 0;  xnodcf : double = 0;  xnodot : double = 0;
    xnodp  : double = 0;
  var
    i                                                   : integer;
    cosuk,sinuk,rfdotk,vx,vy,vz,ux,uy,uz,xmy,xmx,
    cosnok,sinnok,cosik,sinik,rdotk,xinck,xnodek,uk,rk,
    cos2u,sin2u,u,sinu,cosu,betal,rfdot,rdot,r,pl,elsq,
    esine,ecose,epw,temp6,temp5,temp4,cosepw,sinepw,
    capu,ayn,xlt,aynl,xll,axn,xn,beta,xl,e,a,tfour,
    tcube,delm,delomg,templ,tempe,tempa,xnode,tsq,xmp ,
    omega,xnoddf,omgadf,xmdf,x,y,z,xdot,ydot,zdot       : double;
  begin
{ Recover original mean motion (xnodp) and semimajor axis (aodp) }
{ from input elements. }
  if (iflag = 0) then
    goto 100;
```

```
  a1 := Power(xke/xno,tothrd);
  cosio := Cos(xincl);
  theta2 := cosio*cosio;
  x3thm1 := 3*theta2 - 1;
  eosq := eo*eo;
  betao2 := 1 - eosq;
  betao := sqrt(betao2);
  del1 := 1.5*ck2*x3thm1/(a1*a1*betao*betao2);
  ao := a1*(1 - del1*(0.5*tothrd + del1*(1 + 134/81*del1)));
  delo := 1.5*ck2*x3thm1/(ao*ao*betao*betao2);
  xnodp := xno/(1 + delo);
  aodp := ao/(1 - delo);
{ Initialization }
{ For perigee less than 220 kilometers, the isimp flag is set and
  the equations are truncated to linear variation in sqrt a and
  quadratic variation in mean anomaly.  Also, the c3 term, the
  delta omega term, and the delta m term are dropped. }
  isimp := 0;
  if (aodp*(1 - eo)/ae) < (220/xkmper + ae) then
    isimp := 1;
{ For perigee below 156 km, the values of s and qoms2t are altered. }
  s4 := s;
  qoms24 := qoms2t;
  perige := (aodp*(1 - eo) - ae)*xkmper;
  if (perige >= 156) then
    goto 10;
  s4 := perige - 78;
  if (perige > 98) then
    goto 9;
  s4 := 20;
  9:
  qoms24 := Power((120 - s4)*ae/xkmper,4);
  s4 := s4/xkmper + ae;
 10:
  pinvsq := 1/(aodp*aodp*betao2*betao2);
  tsi := 1/(aodp - s4);
  eta := aodp*eo*tsi;
  etasq := eta*eta;
  eeta := eo*eta;
  psisq := abs(1 - etasq);
  coef := qoms24*Power(tsi,4);
  coef1 := coef/Power(psisq,3.5);
  c2 := coef1*xnodp*(aodp*(1 + 1.5*etasq + eeta*(4 + etasq))
      + 0.75*ck2*tsi/psisq*x3thm1*(8 + 3*etasq*(8 + etasq)));
  c1 := bstar*c2,
  sinio := Sin(xincl);
  a3ovk2 := -xj3/ck2*Power(ae,3);
  c3 := coef*tsi*a3ovk2*xnodp*ae*sinio/eo;
  x1mth2 := 1 - theta2;
  c4 := 2*xnodp*coef1*aodp*betao2*(eta*(2 + 0.5*etasq)
      + eo*(0.5 + 2*etasq) - 2*ck2*tsi/(aodp*psisq)
      *(-3*x3thm1*(1 - 2*eeta + etasq*(1.5 - 0.5*eeta))
      + 0.75*x1mth2*(2*etasq - eeta*(1 + etasq))*Cos(2*omegao)));
  c5 := 2*coef1*aodp*betao2*(1 + 2.75*(etasq + eeta) + eeta*etasq);
  theta4 := theta2*theta2;
  temp1 := 3*ck2*pinvsq*xnodp;
  temp2 := temp1*ck2*pinvsq;
  temp3 := 1.25*ck4*pinvsq*pinvsq*xnodp;
  xmdot := xnodp + 0.5*temp1*betao*x3thm1
         + 0.0625*temp2*betao*(13 - 78*theta2 + 137*theta4);
  x1m5th := 1 - 5*theta2;
  omgdot := -0.5*temp1*x1m5th + 0.0625*temp2*(7 - 114*theta2 +395*theta4)
          + temp3*(3 - 36*theta2 + 49*theta4);
  xhdot1 := -temp1*cosio;
  xnodot := xhdot1 + (0.5*temp2*(4 - 19*theta2)
         + 2*temp3*(3 - 7*theta2))*cosio;
  omgcof := bstar*c3*Cos(omegao);
  xmcof := -tothrd*coef*bstar*ae/eeta;
  xnodcf := 3.5*betao2*xhdot1*c1;
  t2cof := 1.5*c1;
  xlcof := 0.125*a3ovk2*sinio*(3 + 5*cosio)/(1 + cosio);
  aycof := 0.25*a3ovk2*sinio;
  delmo := Power(1 + eta*Cos(xmo),3);
  sinmo := Sin(xmo);
  x7thm1 := 7*theta2 - 1;
  if (isimp = 1) then
    goto 90;
  c1sq := c1*c1;
  d2 := 4*aodp*tsi*c1sq;
  temp := d2*tsi*c1/3;
```

```
  d3 := (17*aodp + s4)*temp;
  d4 := 0.5*temp*aodp*tsi*(221*aodp + 31*s4)*c1;
  t3cof := d2 + 2*c1sq;
  t4cof := 0.25*(3*d3 + c1*(12*d2 + 10*c1sq));
  t5cof := 0.2*(3*d4 + 12*c1*d3 + 6*d2*d2 + 15*c1sq*(2*d2 + c1sq));
 90:
  iflag := 0;
{ Update for secular gravity and atmospheric drag. }
100:
  xmdf := xmo + xmdot*tsince;
  omgadf := omegao + omgdot*tsince;
  xnoddf := xnodeo + xnodot*tsince;
  omega := omgadf;
  xmp := xmdf;
  tsq := tsince*tsince;
  xnode := xnoddf + xnodcf*tsq;
  tempa := 1 - c1*tsince;
  tempe := bstar*c4*tsince;
  templ := t2cof*tsq;
  if (isimp = 1) then
    goto 110;
  delomg := omgcof*tsince;
  delm := xmcof*(Power(1 + eta*Cos(xmdf),3) - delmo);
  temp := delomg + delm;
  xmp := xmdf + temp;
  omega := omgadf - temp;
  tcube := tsq*tsince;
  tfour := tsince*tcube;
  tempa := tempa - d2*tsq - d3*tcube - d4*tfour;
  tempe := tempe + bstar*c5*(Sin(xmp) - sinmo);
  templ := templ + t3cof*tcube + tfour*(t4cof + tsince*t5cof);
110:
  a := aodp*Sqr(tempa);
  e := eo - tempe;
  xl := xmp + omega + xnode + xnodp*templ;
  beta := sqrt(1 - e*e);
  xn := xke/Power(a,1.5);
{ Long period periodics }
  axn := e*Cos(omega);
  temp := 1/(a*beta*beta);
  xll := temp*xlcof*axn;
  aynl := temp*aycof;
  xlt := xl + xll;
  ayn := e*Sin(omega) + aynl;
{ Solve Kepler's Equation }
  capu := fmod2p(xlt - xnode);
  temp2 := capu;
  for i := 1 to 10 do
    begin
    sinepw := Sin(temp2);
    cosepw := Cos(temp2);
    temp3 := axn*sinepw;
    temp4 := ayn*cosepw;
    temp5 := axn*cosepw;
    temp6 := ayn*sinepw;
    epw := (capu - temp4 + temp3 - temp2)/(1 - temp5 - temp6) + temp2;
    if (abs(epw - temp2) <= e6a) then
      goto 140;
130:
    temp2 := epw;
    end; {for i}
{ Short period preliminary quantities }
140:
  ecose := temp5 + temp6;
  esine := temp3 - temp4;
  elsq := axn*axn + ayn*ayn;
  temp := 1 - elsq;
  pl := a*temp;
  r := a*(1 - ecose);
  temp1 := 1/r;
  rdot := xke*sqrt(a)*esine*temp1;
  rfdot := xke*sqrt(pl)*temp1;
  temp2 := a*temp1;
  betal := sqrt(temp);
  temp3 := 1/(1 + betal);
  cosu := temp2*(cosepw - axn + ayn*esine*temp3);
  sinu := temp2*(sinepw - ayn - axn*esine*temp3);
  u := actan(sinu,cosu);
  sin2u := 2*sinu*cosu;
  cos2u := 2*cosu*cosu - 1;
```

```
  temp := 1/pl;
  temp1 := ck2*temp;
  temp2 := temp1*temp;
{ Update for short periodics }
  rk := r*(1 - 1.5*temp2*betal*x3thm1) + 0.5*temp1*x1mth2*cos2u;
  uk := u - 0.25*temp2*x7thm1*sin2u;
  xnodek := xnode + 1.5*temp2*cosio*sin2u;
  xinck := xincl + 1.5*temp2*cosio*sinio*cos2u;
  rdotk := rdot - xn*temp1*x1mth2*sin2u;
  rfdotk := rfdot + xn*temp1*(x1mth2*cos2u + 1.5*x3thm1);
{ Orientation vectors }
  sinuk := Sin(uk);
  cosuk := Cos(uk);
  sinik := Sin(xinck);
  cosik := Cos(xinck);
  sinnok := Sin(xnodek);
  cosnok := Cos(xnodek);
  xmx := -sinnok*cosik;
  xmy := cosnok*cosik;
  ux := xmx*sinuk + cosnok*cosuk;
  uy := xmy*sinuk + sinnok*cosuk;
  uz := sinik*sinuk;
  vx := xmx*cosuk - cosnok*sinuk;
  vy := xmy*cosuk - sinnok*sinuk;
  vz := sinik*cosuk;
{ Position and velocity }
  x := rk*ux;  pos[1] := x;
  y := rk*uy;  pos[2] := y;
  z := rk*uz;  pos[3] := z;
  xdot := rdotk*ux + rfdotk*vx;  vel[1] := xdot;
  ydot := rdotk*uy + rfdotk*vy;  vel[2] := ydot;
  zdot := rdotk*uz + rfdotk*vz;  vel[3] := zdot;
  end; {Procedure SGP4}

Procedure Deep(ideep : integer);
  const
    zns    =  1.19459E-5;      c1ss   =  2.9864797E-6;   zes    =  0.01675;
    znl    =  1.5835218E-4;    c1l    =  4.7968065E-7;   zel    =  0.05490;
    zcosis =  0.91744867;      zsinis =  0.39785416;     zsings = -0.98088458;
    zcosgs =  0.1945905;       zcoshs =  1;              zsinhs =  0;
    q22    =  1.7891679E-6;    q31    =  2.1460748E-6;   q33    =  2.2123015E-7;
    g22    =  5.7686396;       g32    =  0.95240898;     g44    =  1.8014998;
    g52    =  1.0508330;       g54    =  4.4108898;      root22 =  1.7891679E-6;
    root32 =  3.7393792E-7;    root44 =  7.3636953E-9;   root52 =  1.1428639E-7;
    root54 =  2.1765803E-9;    thdt   =  4.3752691E-3;
  label
    {dpinit}
    5,10,20,30,40,45,50,55,60,65,70,80,
    {dpsec}
    90,100,105,110,120,125,130,140,150,152,154,160,165,170,175,180,
    {dpper}
    205,210,218,220,230;
const { Typed constants to retain values between ENTRY calls }
    iresfl  : integer = 0;  isynfl  : integer = 0;
    iret    : integer = 0;  iretn   : integer = 0;
    ls      : integer = 0;
    a1      : double = 0;   a2      : double = 0;   a3      : double = 0;
    a4      : double = 0;   a5      : double = 0;   a6      : double = 0;
    a7      : double = 0;   a8      : double = 0;   a9      : double = 0;
    a10     : double = 0;   ainv2   : double = 0;   alfdp   : double = 0;
    aqnv    : double = 0;   atime   : double = 0;   betdp   : double = 0;
    bfact   : double = 0;   c       : double = 0;   cc      : double = 0;
    cosis   : double = 0;   cosok   : double = 0;   cosq    : double = 0;
    ctem    : double = 0;   d2201   : double = 0;   d2211   : double = 0;
    d3210   : double = 0;   d3222   : double = 0;   d4410   : double = 0;
    d4422   : double = 0;   d5220   : double = 0;   d5232   : double = 0;
    d5421   : double = 0;   d5433   : double = 0;   dalf    : double = 0;
    day     : double = 0;   dbet    : double = 0;   del1    : double = 0;
    del2    : double = 0;   del3    : double = 0;   delt    : double = 0;
    dls     : double = 0;   e3      : double = 0;   ee2     : double = 0;
    eoc     : double = 0;   eq      : double = 0;   f2      : double = 0;
    f220    : double = 0;   f221    : double = 0;   f3      : double = 0;
    f311    : double = 0;   f321    : double = 0;   f322    : double = 0;
    f330    : double = 0;   f441    : double = 0;   f442    : double = 0;
    f522    : double = 0;   f523    : double = 0;   f542    : double = 0;
    f543    : double = 0;   fasx2   : double = 0;   fasx4   : double = 0;
    fasx6   : double = 0;   ft      : double = 0;   g200    : double = 0;
    g201    : double = 0;   g211    : double = 0;   g300    : double = 0;
    g310    : double = 0;   g322    : double = 0;   g410    : double = 0;
    g422    : double = 0;   g520    : double = 0;   g521    : double = 0;
    g532    : double = 0;   g533    : double = 0;   gam     : double = 0;
    omegaq  : double = 0;   pe      : double = 0;   pgh     : double = 0;
    ph      : double = 0;   pinc    : double = 0;   pl      : double = 0;
    preep   : double = 0;   s1      : double = 0;   s2      : double = 0;
    s3      : double = 0;   s4      : double = 0;   s5      : double = 0;
```

```
  s6     : double = 0;    s7      : double = 0;    savtsn : double = 0;
  se     : double = 0;    se2     : double = 0;    se3    : double = 0;
  sel    : double = 0;    ses     : double = 0;    sgh    : double = 0;
  sgh2   : double = 0;    sgh3    : double = 0;    sgh4   : double = 0;
  sghl   : double = 0;    sghs    : double = 0;    sh     : double = 0;
  sh2    : double = 0;    sh3     : double = 0;    sh1    : double = 0;
  shs    : double = 0;    si      : double = 0;    si2    : double = 0;
  si3    : double = 0;    sil     : double = 0;    sini2  : double = 0;
  sinis  : double = 0;    sinok   : double = 0;    sinq   : double = 0;
  sinzf  : double = 0;    sis     : double = 0;    sl     : double = 0;
  sl2    : double = 0;    sl3     : double = 0;    sl4    : double = 0;
  sll    : double = 0;    sls     : double = 0;    sse    : double = 0;
  ssg    : double = 0;    ssh     : double = 0;    ssi    : double = 0;
  ssl    : double = 0;    stem    : double = 0;    step2  : double = 0;
  stepn  : double = 0;    stepp   : double = 0;    temp   : double = 0;
  temp1  : double = 0;    thgr    : double = 0;    x1     : double = 0;
  x2     : double = 0;    x2li    : double = 0;    x2omi  : double = 0;
  x3     : double = 0;    x4      : double = 0;    x5     : double = 0;
  x6     : double = 0;    x7      : double = 0;    x8     : double = 0;
  xfact  : double = 0;    xgh2    : double = 0;    xgh3   : double = 0;
  xgh4   : double = 0;    xh2     : double = 0;    xh3    : double = 0;
  xi2    : double = 0;    xi3     : double = 0;    xl     : double = 0;
  xl2    : double = 0;    xl3     : double = 0;    xl4    : double = 0;
  xlamo  : double = 0;    xldot   : double = 0;    xli    : double = 0;
  xls    : double = 0;    xmao    : double = 0;    xnddt  : double = 0;
  xndot  : double = 0;    xni     : double = 0;    xno2   : double = 0;
  xnodce : double = 0;    xnoi    : double = 0;    xnq    : double = 0;
  xomi   : double = 0;    xpidot  : double = 0;    xqncl  : double = 0;
  z1     : double = 0;    z11     : double = 0;    z12    : double = 0;
  z13    : double = 0;    z2      : double = 0;    z21    : double = 0;
  z22    : double = 0;    z23     : double = 0;    z3     : double = 0;
  z31    : double = 0;    z32     : double = 0;    z33    : double = 0;
  zcosg  : double = 0;    zcosgl  : double = 0;    zcosh  : double = 0;
  zcoshl : double = 0;    zcosi   : double = 0;    zcosil : double = 0;
  ze     : double = 0;    zf      : double = 0;    zm     : double = 0;
  zmo    : double = 0;    zmol    : double = 0;    zmos   : double = 0;
  zn     : double = 0;    zsing   : double = 0;    zsingl : double = 0;
  zsinh  : double = 0;    zsinhl  : double = 0;    zsini  : double = 0;
  zsinil : double = 0;    zx      : double = 0;    zy     : double = 0;
begin
case ideep of
  dpinit : begin { Entrance for deep space initialization }
           thgr := Thetag(epoch);
           eq := eo;
           xnq := xnodp;
           aqnv := 1/ao;
           xqncl := xincl;
           xmao := xmo;
           xpidot := omgdt + xnodot;
           sinq := Sin(xnodeo);
           cosq := Cos(xnodeo);
           omegaq := omegao;
  { Initialize lunar solar terms }
        5: day := ds50 + 18261.5;  {Days since 1900 Jan 0.5}
           if (day = preep) then
             Goto 10;
           preep := day;
           xnodce := 4.5236020 - 9.2422029E-4*day;
           stem := Sin(xnodce);
           ctem := Cos(xnodce);
           zcosil := 0.91375164 - 0.03568096*ctem;
           zsinil := Sqrt(1 - zcosil*zcosil);
           zsinhl := 0.089683511*stem/zsinil;
           zcoshl := Sqrt(1 - zsinhl*zsinhl);
           c := 4.7199672 + 0.22997150*day;
           gam := 5.8351514 + 0.0019443680*day;
           zmol := fmod2p(c - gam);
           zx := 0.39785416*stem/zsinil;
           zy := zcoshl*ctem + 0.91744867*zsinhl*stem;
           zx := Actan(zx,zy);
           zx := gam + zx - xnodce;
           zcosgl := Cos(zx);
           zsingl := Sin(zx);
           zmos := 6.2565837 + 0.017201977*day;
           zmos := fmod2p(zmos);
  { Do solar terms }
       10: savtsn := 1E20;
           zcosg := zcosgs;
           zsing := zsings;
           zcosi := zcosis;
           zsini := zsinis;
           zcosh := cosq;
           zsinh := sinq;
           cc := c1ss;
           zn := zns;
           ze := zes;
```

```
        zmo := zmos;
        xnoi := 1/xnq;
        ls := 30; {assign 30 to ls}
    20: a1 := zcosg*zcosh + zsing*zcosi*zsinh;
        a3 := -zsing*zcosh + zcosg*zcosi*zsinh;
        a7 := -zcosg*zsinh + zsing*zcosi*zcosh;
        a8 := zsing*zsini;
        a9 := zsing*zsinh + zcosg*zcosi*zcosh;
        a10 := zcosg*zsini;
        a2 := cosiq*a7 +  siniq*a8;
        a4 := cosiq*a9 +  siniq*a10;
        a5 := -siniq*a7 +  cosiq*a8;
        a6 := -siniq*a9 +  cosiq*a10;
        x1 := a1*cosomo + a2*sinomo;
        x2 := a3*cosomo + a4*sinomo;
        x3 := -a1*sinomo + a2*cosomo;
        x4 := -a3*sinomo + a4*cosomo;
        x5 := a5*sinomo;
        x6 := a6*sinomo;
        x7 := a5*cosomo;
        x8 := a6*cosomo;
        z31 := 12*x1*x1 - 3*x3*x3;
        z32 := 24*x1*x2 - 6*x3*x4;
        z33 := 12*x2*x2 - 3*x4*x4;
        z1 := 3*(a1*a1 + a2*a2) + z31*eqsq;
        z2 := 6*(a1*a3 + a2*a4) + z32*eqsq;
        z3 := 3*(a3*a3 + a4*a4) + z33*eqsq;
        z11 := -6*a1*a5 + eqsq*(-24*x1*x7 - 6*x3*x5);
        z12 := -6*(a1*a6 + a3*a5)
               + eqsq*(-24*(x2*x7 + x1*x8) - 6*(x3*x6 + x4*x5));
        z13 := -6*a3*a6 + eqsq*(-24*x2*x8 - 6*x4*x6);
        z21 := 6*a2*a5 + eqsq*(24*x1*x5 - 6*x3*x7);
        z22 := 6*(a4*a5 + a2*a6)
               + eqsq*(24*(x2*x5 + x1*x6) - 6*(x4*x7 + x3*x8));
        z23 := 6*a4*a6 + eqsq*(24*x2*x6 - 6*x4*x8);
        z1 := z1 + z1 + bsq*z31;
        z2 := z2 + z2 + bsq*z32;
        z3 := z3 + z3 + bsq*z33;
        s3 := cc*xnoi;
        s2 := -0.5*s3/rteqsq;
        s4 := s3*rteqsq;
        s1 := -15*eq*s4;
        s5 := x1*x3 + x2*x4;
        s6 := x2*x3 + x1*x4;
        s7 := x2*x4 - x1*x3;
        se := s1*zn*s5;
        si := s2*zn*(z11 + z13);
        sl := -zn*s3*(z1 + z3 - 14 - 6*eqsq);
        sgh := s4*zn*(z31 + z33 - 6);
        sh := -zn*s2*(z21 + z23);
        if (xqncl < 5.2359877E-2) then
          sh := 0;
        ee2 := 2*s1*s6;
        e3 := 2*s1*s7;
        xi2 := 2*s2*z12;
        xi3 := 2*s2*(z13 - z11);
        xl2 := -2*s3*z2;
        xl3 := -2*s3*(z3 - z1);
        xl4 := -2*s3*(-21 - 9*eqsq)*ze;
        xgh2 := 2*s4*z32;
        xgh3 := 2*s4*(z33 - z31);
        xgh4 := -18*s4*ze;
        xh2 := -2*s2*z22;
        xh3 := -2*s2*(z23 - z21);
        case ls of
          30 : Goto 30;
          40 : Goto 40;
        else
             Halt;
        end; {case}
{ Do lunar terms }
    30: sse := se;
        ssi := si;
        ssl := sl;
        ssh := sh/siniq;
        ssg := sgh - cosiq*ssh;
        se2 := ee2;
        si2 := xi2;
        sl2 := xl2;
        sgh2 := xgh2;
        sh2 := xh2;
        se3 := e3;
        si3 := xi3;
        sl3 := xl3;
        sgh3 := xgh3;
```

```
        sh3 := xh3;
        sl4 := xl4;
        sgh4 := xgh4;
        zcosg := zcosgl;
        zsing := zsingl;
        zcosi := zcosil;
        zsini := zsinil;
        zcosh := zcoshl*cosq + zsinhl*sinq;
        zsinh := sinq*zcoshl - cosq*zsinhl;
        zn := znl;
        cc := c1l;
        ze := zel;
        zmo := zmol;
        ls := 40; {assign 40 to ls}
        Goto 20;
    40: sse := sse + se;
        ssi := ssi + si;
        ssl := ssl + sl;
        ssg := ssg + sgh - cosiq/siniq*sh;
        ssh := ssh + sh/siniq;
{ Geopotential resonance initialization for 12 hour orbits }
        iresfl := 0;
        isynfl := 0;
        if (xnq < 0.0052359877) and (xnq > 0.0034906585) then
          Goto 70;
        if (xnq < 8.26E-3) or (xnq > 9.24E-3) then
          exit;
        if (eq < 0.5) then
          exit;
        iresfl := 1;
        eoc := eq*eqsq;
        g201 := -0.306 - (eq - 0.64)*0.440;
        if (eq > 0.65) then
          Goto 45;
        g211 := 3.616 - 13.247*eq + 16.290*eqsq;
        g310 := -19.302 + 117.390*eq - 228.419*eqsq + 156.591*eoc;
        g322 := -18.9068 + 109.7927*eq - 214.6334*eqsq + 146.5816*eoc;
        g410 := -41.122 + 242.694*eq - 471.094*eqsq + 313.953*eoc;
        g422 := -146.407 + 841.880*eq - 1629.014*eqsq + 1083.435*eoc;
        g520 := -532.114 + 3017.977*eq - 5740*eqsq + 3708.276*eoc;
        Goto 55;
    45: g211 := -72.099 + 331.819*eq - 508.738*eqsq + 266.724*eoc;
        g310 := -346.844 + 1582.851*eq - 2415.925*eqsq + 1246.113*eoc;
        g322 := -342.585 + 1554.908*eq - 2366.899*eqsq + 1215.972*eoc;
        g410 := -1052.797 + 4758.686*eq - 7193.992*eqsq + 3651.957*eoc;
        g422 := -3581.69 + 16178.11*eq - 24462.77*eqsq + 12422.52*eoc;
        if (eq > 0.715) then
          Goto 50;
        g520 := 1464.74 - 4664.75*eq + 3763.64*eqsq;
        Goto 55;
    50: g520 := -5149.66 + 29936.92*eq - 54087.36*eqsq + 31324.56*eoc;
    55: if (eq >= (0.7)) then
          Goto 60;
        g533 := -919.2277 + 4988.61*eq - 9064.77*eqsq + 5542.21*eoc;
        g521 := -822.71072 + 4568.6173*eq - 8491.4146*eqsq + 5337.524*eoc;
        g532 := -853.666 + 4690.25*eq - 8624.77*eqsq + 5341.4*eoc;
        Goto 65;
    60: g533 := -37995.78 + 161616.52*eq - 229838.2*eqsq + 109377.94*eoc;
        g521 := -51752.104 + 218913.95*eq - 309468.16*eqsq + 146349.42*eoc;
        g532 := -40023.88 + 170470.89*eq - 242699.48*eqsq + 115605.82*eoc;
    65: sini2 := siniq*siniq;
        f220 := 0.75*(1 + 2*cosiq + cosq2);
        f221 := 1.5*sini2;
        f321 := 1.875*siniq*(1 - 2*cosiq - 3*cosq2);
        f322 := -1.875*siniq*(1 + 2*cosiq - 3*cosq2);
        f441 := 35*sini2*f220;
        f442 := 39.3750*sini2*sini2;
        f522 := 9.84375*siniq*(sini2*(1 - 2*cosiq - 5*cosq2)
              + 0.33333333*(-2 + 4*cosiq + 6*cosq2));
        f523 := siniq*(4.92187512*sini2*(-2 - 4*cosiq + 10*cosq2)
              + 6.56250012*(1 + 2*cosiq - 3*cosq2));
        f542 := 29.53125*siniq*(2 - 8*cosiq + cosq2*(-12 + 8*cosiq + 10*cosq2));
        f543 := 29.53125*siniq*(-2 - 8*cosiq + cosq2*(12 + 8*cosiq - 10*cosq2));
        xno2 := xnq*xnq;
        ainv2 := aqnv*aqnv;
        temp1 := 3*xno2*ainv2;
        temp := temp1*root22;
        d2201 := temp*f220*g201;
        d2211 := temp*f221*g211;
        temp1 := temp1*aqnv;
        temp := temp1*root32;
        d3210 := temp*f321*g310;
```

```
        d3222 := temp*f322*g322;
        temp1 := temp1*aqnv;
        temp := 2*temp1*root44;
        d4410 := temp*f441*g410;
        d4422 := temp*f442*g422;
        temp1 := temp1*aqnv;
        temp := temp1*root52;
        d5220 := temp*f522*g520;
        d5232 := temp*f523*g532;
        temp := 2*temp1*root54;
        d5421 := temp*f542*g521;
        d5433 := temp*f543*g533;
        xlamo := xmao + xnodeo + xnodeo - thgr - thgr;
        bfact := xlldot + xnodot + xnodot - thdt - thdt;
        bfact := bfact + ssl + ssh + ssh;
        Goto 80;
{ Synchronous resonance terms initialization }
    70: iresfl := 1;
        isynfl := 1;
        g200 := 1 + eqsq*(-2.5 + 0.8125*eqsq);
        g310 := 1 + 2*eqsq;
        g300 := 1 + eqsq*(-6 + 6.60937*eqsq);
        f220 := 0.75*(1 + cosiq)*(1 + cosiq);
        f311 := 0.9375*siniq*siniq*(1 + 3*cosiq) - 0.75*(1 + cosiq);
        f330 := 1 + cosiq;
        f330 := 1.875*f330*f330*f330;
        del1 := 3*xnq*xnq*aqnv*aqnv;
        del2 := 2*del1*f220*g200*q22;
        del3 := 3*del1*f330*g300*q33*aqnv;
        del1 := del1*f311*g310*q31*aqnv;
        fasx2 := 0.13130908;
        fasx4 := 2.8843198;
        fasx6 := 0.37448087;
        xlamo := xmao + xnodeo + omegao - thgr;
        bfact := xlldot + xpidot - thdt;
        bfact := bfact + ssl + ssg + ssh;
    80: xfact := bfact - xnq;
{ Initialize integrator }
        xli := xlamo;
        xni := xnq;
        atime := 0;
        stepp := 720;
        stepn := -720;
        step2 := 259200;
        end; {dpinit}
dpsec  : begin { Entrance for deep space secular effects }
        xll := xll + ssl*t;
        omgasm := omgasm + ssg*t;
        xnodes := xnodes + ssh*t;
        _em := eo + sse*t;
        xinc := xincl + ssi*t;
        if (xinc >= 0) then
          Goto 90;
        xinc := -xinc;
        xnodes := xnodes  +  pi;
        omgasm := omgasm - pi;
    90: if (iresfl = 0) then
          exit;
   100: if (atime = 0) then
          Goto 170;
        if (t >= 0) and (atime < 0) then
          Goto 170;
        if (t < 0) and (atime >= 0) then
          Goto 170;
   105: if (Abs(t) >= Abs(atime)) then
          Goto 120;
        delt := stepp;
        if (t >= 0) then
          delt := stepn;
   110: iret := 100; {assign 100 to iret}
        Goto 160;
   120: delt := stepn;
        if (t > 0) then
          delt := stepp;
   125: if (Abs(t - atime) < stepp) then
          Goto 130;
        iret := 125; {assign 125 to iret}
        Goto 160;
   130: ft := t - atime;
        iretn := 140; {assign 140 to iretn}
        Goto 150;
   140: xn := xni + xndo       + xnddt*ft*ft*0.5;
        xl := xli + :l         xndot*ft*ft*0.5;
        temp := -xnodes        + t*thdt;
```

```
        xll := xl - omgasm + temp;
        if (isynfl = 0) then
          xll := xl + temp + temp;
        exit;
{ Dot terms calculated }
    150: if (isynfl = 0) then
          Goto 152;
        xndot := del1*Sin(xli - fasx2) + del2*Sin(2*(xli - fasx4))
               + del3*Sin(3*(xli - fasx6));
        xnddt := del1*Cos(xli - fasx2)
               + 2*del2*Cos(2*(xli - fasx4))
               + 3*del3*Cos(3*(xli - fasx6));
        Goto 154;
    152: xomi := omegaq + omgdt*atime;
        x2omi := xomi + xomi;
        x2li := xli + xli;
        xndot := d2201*Sin(x2omi + xli - g22)
               + d2211*Sin(xli - g22)
               + d3210*Sin(xomi + xli - g32)
               + d3222*Sin(-xomi + xli - g32)
               + d4410*Sin(x2omi + x2li - g44)
               + d4422*Sin(x2li - g44)
               + d5220*Sin(xomi + xli - g52)
               + d5232*Sin(-xomi + xli - g52)
               + d5421*Sin(xomi + x2li - g54)
               + d5433*Sin(-xomi + x2li - g54);
        xnddt := d2201*Cos(x2omi + xli - g22)
               + d2211*Cos(xli - g22)
               + d3210*Cos(xomi + xli - g32)
               + d3222*Cos(-xomi + xli - g32)
               + d5220*Cos(xomi + xli - g52)
               + d5232*Cos(-xomi + xli - g52)
               + 2*(d4410*Cos(x2omi + x2li - g44)
               + d4422*Cos(x2li - g44)
               + d5421*Cos(xomi + x2li - g54)
               + d5433*Cos(-xomi + x2li - g54));
    154: xldot := xni + xfact;
        xnddt := xnddt*xldot;
        case iretn of
          140 : Goto 140;
          165 : Goto 165;
        else
                Halt;
        end; {case}
{ Integrator }
    160: iretn := 165; {assign 165 to iretn}
        Goto 150;
    165: xli := xli + xldot*delt + xndot*step2;
        xni := xni + xndot*delt + xnddt*step2;
        atime := atime + delt;
        case iret of
          100 : Goto 100;
          125 : Goto 125;
        else
                Halt;
        end; {case}
{ Epoch restart }
    170: if (t >= 0) then
          Goto 175;
        delt := stepn;
        Goto 180;
    175: delt := stepp;
    180: atime := 0;
        xni := xnq;
        xli := xlamo;
        Goto 125;
        end; {dpsec}
dpper  : begin { Entrance for lunar-solar periodics }
        sinis := Sin(xinc);
        cosis := Cos(xinc);
        if (Abs(savtsn - t) < 30) then
          Goto 210;
        savtsn := t;
        zm := zmos + zns*t;
    205: zf := zm + 2*zes*Sin(zm);
        sinzf := Sin(zf);
        f2 := 0.5*sinzf*sinzf - 0.25;
        f3 := -0.5*sinzf*Cos(zf);
        ses := se2*f2 + se3*f3;
        sis := si2*f2 + si3*f3;
        sls := sl2*f2 + sl3*f3 + sl4*sinzf;
        sghs := sgh2*f2 + sgh3*f3 + sgh4*sinzf;
        shs := sh2*f2 + sh3*f3;
        zm := zmol + znl*t;
```

```
          zf := zm + 2*zel*Sin(zm);
          sinzf := Sin(zf);
          f2 := 0.5*sinzf*sinzf - 0.25;
          f3 := -0.5*sinzf*Cos(zf);
          sel := ee2*f2 + e3*f3;
          sil := xi2*f2 + xi3*f3;
          sll := xl2*f2 + xl3*f3 + xl4*sinzf;
          sghl := xgh2*f2 + xgh3*f3 + xgh4*sinzf;
          shl := xh2*f2 + xh3*f3;
          pe := ses + sel;
          pinc := sis + sil;
          pl := sls + sll;
     210: pgh := sghs + sghl;
          ph := shs + shl;
          xinc := xinc + pinc;
          _em := _em + pe;
          if (xqncl < 0.2) then
            Goto 220;
          Goto 218;
   { Apply periodics directly }
     218: ph := ph/siniq;
          pgh := pgh - cosiq*ph;
          omgasm := omgasm + pgh;
          xnodes := xnodes + ph;
          xll := xll + pl;
          Goto 230;
   { Apply periodics with Lyddane modification }
     220: sinok := Sin(xnodes);
          cosok := Cos(xnodes);
          alfdp := sinis*sinok;
          betdp := sinis*cosok;
          dalf := ph*cosok + pinc*cosis*sinok;
          dbet := -ph*sinok + pinc*cosis*cosok;
          alfdp := alfdp + dalf;
          betdp := betdp + dbet;
          xls := xll + omgasm + cosis*xnodes;
          dls := pl + pgh - pinc*xnodes*sinis;
          xls := xls + dls;
          xnodes := Actan(alfdp,betdp);
          xll := xll + pl;
          omgasm := xls - xll - Cos(xinc)*xnodes;
     230: end; {dpper}
   end; {case}
  end; {Procedure Deep}

Procedure Call_dpinit(var eosq,sinio,cosio,betao,aodp,theta2,
                          sing,cosg,betao2,xmdot,omgdot,
                          xnodott,xnodpp : double);
  begin
  eqsq   := eosq;    siniq  := sinio;  cosiq  := cosio;   rteqsq := betao;
  ao     := aodp;    cosq2  := theta2; sinomo := sing;    cosomo := cosg;
  bsq    := betao2;  xlldot := xmdot;  omgdt  := omgdot;  xnodot := xnodott;
  xnodp  := xnodpp;
  Deep(1);
  eosq   := eqsq;    sinio  := siniq;  cosio  := cosiq;   betao  := rteqsq;
  aodp   := ao;      theta2 := cosq2;  sing   := sinomo;  cosg   := cosomo;
  betao2 := bsq;     xmdot  := xlldot; omgdot := omgdt;   xnodott := xnodot;
  xnodpp  := xnodp;
  end; {Procedure Call_dpinit}

Procedure Call_dpsec(var xmdf,omgadf,xnode,emm,xincc,xnn,tsince : double);
  begin
  xll    := xmdf;    omgasm := omgadf; xnodes := xnode;   {_em     := emm;
  xinc   := xincc;}  xn     := xnn;    t      := tsince;
  Deep(2);
  xmdf   := xll;     omgadf := omgasm; xnode  := xnodes;  emm    := _em;
  xincc  := xinc;    xnn    := xn;     tsince := t;
  end. {Procedure Call_dpsec}

Procedure Call_dpper(var e,xincc,omgadf,xnode,xmam : double);
  begin
  _em     := e,       xinc   := xincc;   omgasm := omgadf;  xnodes := xnode;
  xll     := xmam;
  Deep(3);
  e       := _em;     xincc  := xinc;    omgadf := omgasm;  xnode  := xnodes;
  xmam    := xll;
  end; {Procedure Call_dpper}

Procedure SDP4(tsince : double;
           var iflag : integer;
          var pos,vel : vector);
  label
```

```
    9,10,90,100,130,140;
  const
    a1     : double = 0;  a3ovk2 : double = 0;  ao     : double = 0;
    aodp   : double = 0;  aycof  : double = 0;  betao  : double = 0;
    betao2 : double = 0;  c1     : double = 0;  c2     : double = 0;
    c4     : double = 0;  coef   : double = 0;  coef1  : double = 0;
    cosg   : double = 0;  cosio  : double = 0;  del1   : double = 0;
    delo   : double = 0;  eeta   : double = 0;  eosq   : double = 0;
    eta    : double = 0;  etasq  : double = 0;  omgdot : double = 0;
    perige : double = 0;  pinvsq : double = 0;  psisq  : double = 0;
    qoms24 : double = 0;  s4     : double = 0;  sing   : double = 0;
    sinio  : double = 0;  t2cof  : double = 0;  temp1  : double = 0;
    temp2  : double = 0;  temp3  : double = 0;  theta2 : double = 0;
    theta4 : double = 0;  tsi    : double = 0;  x1m5th : double = 0;
    x1mth2 : double = 0;  x3thm1 : double = 0;  x7thm1 : double = 0;
    xhdot1 : double = 0;  xlcof  : double = 0;  xmdot  : double = 0;
    xnodcf : double = 0;  xnodot : double = 0;  xnodp  : double = 0;
  var
    i                                                    : integer;
    a,axn,ayn,aynl,beta,betal,capu,cos2u,cosepw,cosik,
    cosnok,cosu,cosuk,e,ecose,elsq,em,epw,esine,omgadf,
    pl,r,rdot,rdotk,rfdot,rfdotk,rk,sin2u,sinepw,sinik,
    sinnok,sinu,sinuk,temp,temp4,temp5,temp6,tempa,
    tempe,templ,tsq,u,uk,ux,uy,uz,vx,vy,vz,xinc,xinck,
    xl,xll,xlt,xmam,xmdf,xmx,xmy,xn,xnoddf,xnode,xnodek,
    x,y,z,xdot,ydot,zdot                                 : double;
  begin
  if (iflag = 0) then
    goto 100;
{ Recover original mean motion (xnodp) and semimajor axis (aodp) }
{ from input elements. }
  a1 := Power(xke/xno,tothrd);
  cosio := cos(xincl);
  theta2 := cosio*cosio;
  x3thm1 := 3*theta2 - 1;
  eosq := eo*eo;
  betao2 := 1 - eosq;
  betao := sqrt(betao2);
  del1 := 1.5*ck2*x3thm1/(a1*a1*betao*betao2);
  ao := a1*(1 - del1*(0.5*tothrd + del1*(1 + 134/81*del1)));
  delo := 1.5*ck2*x3thm1/(ao*ao*betao*betao2);
  xnodp := xno/(1 + delo);
  aodp := ao/(1 - delo);
{ Initialization }
{ For perigee below 156 km, the values of s and qoms2t are altered. }
  s4 := s;
  qoms24 := qoms2t;
  perige := (aodp*(1 - eo) - ae)*xkmper;
  if (perige >= 156) then goto 10;
  s4 := perige - 78;
  if (perige > 98) then goto 9;
  s4 := 20;
  9:
  qoms24 := Power((120 - s4)*ae/xkmper,4);
  s4 := s4/xkmper + ae;
 10:
  pinvsq := 1/(aodp*aodp*betao2*betao2);
  sing := sin(omegao);
  cosg := cos(omegao);
  tsi := 1/(aodp - s4);
  eta := aodp*eo*tsi;
  etasq := eta*eta;
  eeta := eo*eta;
  psisq := abs(1 - etasq);
  coef := qoms24*Power(tsi,4);
  coef1 := coef/Power(psisq,3.5);
  c2 := coef1*xnodp*(aodp*(1 + 1.5*etasq + eeta*(4 + etasq))
      + 0.75*ck2*tsi/psisq*x3thm1*(8 + 3*etasq*(8 + etasq)));
  c1 := bstar*c2;
  sinio := sin(xincl);
  a3ovk2 := -xj3/ck2*Power(ae,3);
  x1mth2 := 1 - theta2;
  c4 := 2*xnodp*coef1*aodp*betao2*(eta*(2 + 0.5*etasq)
      + eo*(0.5 + 2*etasq) - 2*ck2*tsi/(aodp*psisq)
      *(-3*x3thm1*(1 - 2*eeta + etasq*(1.5 - 0.5*eeta))
      + 0.75*x1mth2*(2*etasq - eeta*(1 + etasq))*cos(2*omegao)));
  theta4 := theta2*theta2;
  temp1 := 3*ck2*pinvsq*xnodp;
  temp2 := temp1*ck2*pinvsq;
  temp3 := 1.25*ck4*pinvsq*pinvsq*xnodp;
  xmdot := xnodp + 0.5*temp1*betao*x3thm1
```

```
        + 0.0625*temp2*betao*(13 - 78*theta2 + 137*theta4);
  x1m5th := 1 - 5*theta2;
  omgdot := -0.5*temp1*x1m5th + 0.0625*temp2*(7 - 114*theta2 + 395*theta4)
          + temp3*(3 - 36*theta2 + 49*theta4);
  xhdot1 := -temp1*cosio;
  xnodot := xhdot1 + (0.5*temp2*(4 - 19*theta2)
          + 2*temp3*(3 - 7*theta2))*cosio;
  xnodcf := 3.5*betao2*xhdot1*c1;
  t2cof := 1.5*c1;
  xlcof := 0.125*a3ovk2*sinio*(3 + 5*cosio)/(1 + cosio);
  aycof := 0.25*a3ovk2*sinio;
  x7thm1 := 7*theta2 - 1;
 90:
  iflag := 0;
  Call_dpinit(eosq,sinio,cosio,betao,aodp,theta2,sing,cosg,
              betao2,xmdot,omgdot,xnodot,xnodp);
{ Update for secular gravity and atmospheric drag }
100:
  xmdf := xmo + xmdot*tsince;
  omgadf := omegao + omgdot*tsince;
  xnoddf := xnodeo + xnodot*tsince;
  tsq := tsince*tsince;
  xnode := xnoddf + xnodcf*tsq;
  tempa := 1 - c1*tsince;
  tempe := bstar*c4*tsince;
  templ := t2cof*tsq;
  xn := xnodp;
  Call_dpsec(xmdf,omgadf,xnode,em,xinc,xn,tsince);
  a := Power(xke/xn,tothrd)*Sqr(tempa);
  e := em - tempe;
  xmam := xmdf + xnodp*templ;
  Call_dpper(e,xinc,omgadf,xnode,xmam);
  xl := xmam + omgadf + xnode;
  beta := sqrt(1 - e*e);
  xn := xke/Power(a,1.5);
{ Long period periodics }
  axn := e*cos(omgadf);
  temp := 1/(a*beta*beta);
  xll := temp*xlcof*axn;
  aynl := temp*aycof;
  xlt := xl + xll;
  ayn := e*sin(omgadf) + aynl;
{ Solve Kepler's Equation }
  capu := fmod2p(xlt - xnode);
  temp2 := capu;
  for i := 1 to 10 do
    begin
    sinepw := sin(temp2);
    cosepw := cos(temp2);
    temp3 := axn*sinepw;
    temp4 := ayn*cosepw;
    temp5 := axn*cosepw;
    temp6 := ayn*sinepw;
    epw := (capu - temp4 + temp3 - temp2)/(1 - temp5 - temp6) + temp2;
    if (abs(epw - temp2) <= e6a) then goto 140;
130
    temp2 := epw;
    end; {for i}
{ Short period preliminary quantities }
140:
  ecose := temp5 + temp6;
  esine := temp3 - temp4;
  elsq := axn*axn + ayn*ayn;
  temp := 1 - elsq;
  pl := a*temp;
  r := a*(1 - ecose);
  temp1 := 1/r;
  rdot := xke*sqrt(a)*esine*temp1;
  rfdot := xke*sqrt(pl)*temp1;
  temp2 := a*temp1;
  betal := sqrt(temp);
  temp3 := 1/(1 + betal);
  cosu := temp2*(cosepw - axn + ayn*esine*temp3);
  sinu := temp2*(sinepw - ayn - axn*esine*temp3);
  u := actan(sinu,cosu);
  sin2u := 2*sinu*cosu;
  cos2u := 2*cosu*cosu - 1;
  temp := 1/pl;
  temp1 := ck2*temp;
  temp2 := temp1*temp;
```

```
{ Update for short periodics }
  rk := r*(1 - 1.5*temp2*betal*x3thm1) + 0.5*temp1*x1mth2*cos2u;
  uk := u - 0.25*temp2*x7thm1*sin2u;
  xnodek := xnode + 1.5*temp2*cosio*sin2u;
  xinck := xinc + 1.5*temp2*cosio*sinio*cos2u;
  rdotk := rdot - xn*temp1*x1mth2*sin2u;
  rfdotk := rfdot + xn*temp1*(x1mth2*cos2u + 1.5*x3thm1);
{ Orientation vectors }
  sinuk := sin(uk);
  cosuk := cos(uk);
  sinik := sin(xinck);
  cosik := cos(xinck);
  sinnok := sin(xnodek);
  cosnok := cos(xnodek);
  xmx := -sinnok*cosik;
  xmy := cosnok*cosik;
  ux := xmx*sinuk + cosnok*cosuk;
  uy := xmy*sinuk + sinnok*cosuk;
  uz := sinik*sinuk;
  vx := xmx*cosuk - cosnok*sinuk;
  vy := xmy*cosuk - sinnok*sinuk;
  vz := sinik*cosuk;
{ Position and velocity }
  x := rk*ux;  pos[1] := x;
  y := rk*uy;  pos[2] := y;
  z := rk*uz;  pos[3] := z;
  xdot := rdotk*ux + rfdotk*vx;  vel[1] := xdot;
  ydot := rdotk*uy + rfdotk*vy;  vel[2] := ydot;
  zdot := rdotk*uz + rfdotk*vz;  vel[3] := zdot;
  end; {Procedure SDP4}

Procedure SGP(time : double;
        var pos,vel : vector);
  var
    tsince : double;
  begin
  tsince := (time - julian_epoch) * xmnpda;
  if ideep = 0 then
    SGP4(tsince,iflag,pos,vel)
  else
    SDP4(tsince,iflag,pos,vel);
  end; {Procedure SGP}

  begin
  Define_Derived_Constants;
end.
```

B.2 SGP_INIT Unit Source Code Listing

```
Unit SGP_Init;
{           Author:  Dr TS Kelso }
{ Original Version:  1992 Sep 01 }
{ Current Revision:  1992 Sep 13 }
{          Version:  1.10 }
{        Copyright:  1992, All Rights Reserved }
{$N+}

INTERFACE

const
  max_sats = 250;

type
  line_data = string[69];
  two_line  = array [1..2] of line_data;

var
  visible                 : boolean;
  epoch                   : double;
  catnr,elset             : string;
  obs_name                : string[25];
  selected                : array [1..max_sats] of boolean;
  sat_name                : array [1..max_sats] of string[22];
  sat_data                : array [1..max_sats] of two_line;
  data_drive,data_dir,
  work_drive,work_dir     : string;

Procedure Program_Initialize(program_name : string);
Procedure Program_End;

IMPLEMENTATION
  Uses CRT,Support;

Procedure Program_Initialize(program_name : string);
  var
    space,lines : byte;
    line,fn     : string;
    fi          : text;
  begin
{ Input header file describing program, 22 lines by 79 columns maximum }
  ClrScr;
  fn := program_name + '.HDR';
  if File_Exists(fn) then
    begin
    Assign(fi,fn);
    Reset(fi);
    lines := 0;
    repeat
      lines := lines + 1;
      Readln(fi,line);
      Writeln(line);
    until EOF(fi) or (lines = 22);
    Close(fi);
    end; {if}
{ Input directory configuration file }
  fn := program_name + '.CFG';
  if File_Exists(fn) then
    begin
    Assign(fi,fn);
    Reset(fi);
    Readln(fi,data_drive);
    if data_drive <> '' then
      begin
      data_drive[1] := UpCase(data_drive[1]);
      if data_drive[1] in ['A'..'Z'] then
        data_drive := data_drive[1] + ':'
      else
        data_drive := '';
      end; {if}
    Readln(fi,data_dir);
    if data_dir <> '' then
      begin
      space := Pos(' ',data_dir);
      if space > 0 then
        data_dir := Copy(data_dir,1,space-1);
      if data_dir[Length(data_dir)] <> '\' then
        data_dir := data_dir + '\';
      end; {if}
    Readln(fi,work_drive);
    if work_drive <> '' then
      begin
```

```
      work_drive[1] := UpCase(work_drive[1]);
      if work_drive[1] in ['A'..'Z'] then
        work_drive := work_drive[1] + ':'
      else
        work_drive := '';
      end; {if}
    Readln(fi,work_dir);
    if work_dir <> '' then
      begin
      space := Pos(' ',work_dir);
      if space > 0 then
        work_dir := Copy(work_dir,1,space-1);
      if work_dir[Length(work_dir)] <> '\' then
        work_dir := work_dir + '\';
      end; {if}
    Close(fi);
    end {if}
  else
    begin
    data_drive := '';
    data_dir   := '';
    work_drive := '';
    work_dir   := '';
    end; {else}
  end; {Procedure Initialize}

Procedure Program_End;
  begin
  GotoXY(1,24);
  Cursor_On;
  end; {Procedure Program_End}

end.
```

B.3 SGP_INTF Unit Source Code Listing

```
Unit SGP_Intf;
{           Author:  Dr TS Kelso }
{ Original Version:  1992 Sep 03 }
{ Current Revision:  1992 Sep 13 }
{          Version:  1.02 }
{        Copyright:  1992, All Rights Reserved }
{$N+}

INTERFACE

const
  ae        = 1;
  tothrd    = 2/3;
  xkmper    = 6378.135;        {Earth equatorial radius - kilometers (WGS '72)}
  f         = 1/298.26;        {Earth flattening (WGS '72)}
  ge        = 398600.8;        {Earth gravitational constant (WGS '72)}
  J2        = 1.0826158E-3;    {J2 harmonic (WGS '72)}
  J3        = -2.53881E-6;     {J3 harmonic (WGS '72)}
  J4        = -1.65597E-6;     {J4 harmonic (WGS '72)}
  ck2       = J2/2;
  ck4       = -3*J4/8;
  xj3       = J3;
  qo        = ae + 120/xkmper;
  s         = ae + 78/xkmper;
  e6a       = 1E-6;
  dpinit    = 1;               {Deep-space initialization code}
  dpsec     = 2;               {Deep-space secular code}
  dpper     = 3;               {Deep-space periodic code}

var
  iflag,ideep                  : integer;
  xmo,xnodeo,omegao,eo,xincl,
  xno,xndt2o,xndd6o,bstar,
  julian_epoch,xke             : double;

IMPLEMENTATION

end.
```

B.4 SGP_MATH Unit Source Code Listing

```
Unit SGP_Math;
{           Author:  Dr TS Kelso }
{ Original Version:  1991 Oct 30 }
{ Current Revision:  1992 Sep 28 }
{          Version:  1.30 }
{        Copyright:  1991-1992, All Rights Reserved }
{$N+}

INTERFACE

type
  vector = array [1..4] of double;

const
  twopi = 2 * pi;
  zero : vector = (0,0,0,0);

Function Sign(arg : double) : shortint;
Function Cube(arg : double) : double;
Function Power(arg,pwr : double) : double;
Function Radians(arg : double) : double;
Function Degrees(arg : double) : double;
Function Tan(arg : double) : double;
Function ArcSin(arg : double) : double;
Function ArcCos(arg : double) : double;
Function Modulus(arg1,arg2 : double) : double;
Function Fmod2p(arg : double) : double;
Function AcTan(sinx,cosx : double) : double;
Function Dot(v1,v2 : vector) : double;
Procedure Magnitude(var v : vector);
Procedure Cross(v1,v2 : vector; var v3 : vector);

IMPLEMENTATION

Function Sign(arg : double) : shortint;
  begin
  if arg > 0 then
    Sign := 1
  else if arg < 0 then
    Sign := -1
  else
    Sign := 0;
  end; {Function Sign}

Function Cube(arg : double) : double;
  begin
  Cube := arg*Sqr(arg);
  end; {Function Cube}

Function Power(arg,pwr : double) : double;
  begin
  if arg > 0 then
    Power := Exp(pwr*Ln(arg))
  else
    Writeln(output,'Invalid argument in Function Power!');
  end; {Function Power}

Function Radians(arg : double) : double;
  begin
  Radians := arg*pi/180;
  end; {Function Radians}

Function Degrees(arg : double) : double;
  begin
  Degrees := arg*180/pi;
  end; {Function Degrees}

Function Tan(arg : double) : double;
  begin
  Tan := Sin(arg)/Cos(arg);
  end; {Function Tan}

Function ArcSin(arg : double) : double;
  begin
  ArcSin := ArcTan(arg/Sqrt(1-Sqr(arg)));
  end; {Function ArcSin}

Function ArcCos(arg : double) : double;
  begin
  ArcCos := pi/2 - ArcSin(arg);
```

```
  end; {Function ArcCos}

Function Modulus(arg1,arg2 : double) : double;
  var
    modu : double;
  begin
  modu := arg1 - Trunc(arg1/arg2) * arg2;
  if modu >= 0 then
    Modulus := modu
  else
    Modulus := modu + arg2;
  end; {Function Modulus}

Function Fmod2p(arg : double) : double;
  begin
  Fmod2p := Modulus(arg,twopi);
  end; {Function Fmod2p}

Function AcTan(sinx,cosx : double) : double;
  begin
  if cosx = 0 then
    if sinx > 0 then
      Actan := pi/2
    else
      Actan := 3*pi/2
  else if cosx > 0 then
    Actan := ArcTan(sinx/cosx)
  else
    Actan := pi + ArcTan(sinx/cosx);
  end; {Function Actan}

Function Dot(v1,v2 : vector) : double;
  begin
  Dot := v1[1]*v2[1] + v1[2]*v2[2] + v1[3]*v2[3];
  end;  {Function Dot}

Procedure Magnitude(var v : vector);
  begin
  v[4] := Sqrt(Sqr(v[1]) + Sqr(v[2]) + Sqr(v[3]));
  end; {Procedure Magnitude}

Procedure Cross(v1,v2 : vector; var v3 : vector);
  begin
  v3[1] := v1[2]*v2[3] - v1[3]*v2[2];
  v3[2] := v1[3]*v2[1] - v1[1]*v2[3];
  v3[3] := v1[1]*v2[2] - v1[2]*v2[1];
  end; {Procedure Cross}

end.
```

B.5 SGP_TIME Unit Source Code Listing

```
Unit SGP_Time;
{            Author:  Dr TS Kelso }
{ Original Version:  1992 Jun 02 }
{ Current Revision:  1992 Sep 28 }
{           Version:  1.50 }
{         Copyright:  1992, All Rights Reserved }
{$N+}
INTERFACE
  Uses SGP_Math;

type
  clock_time = string[12];
  date       = string[11];

const
  xmnpda   =  1440.0;        {Minutes per day}
  secday   = 86400.0;        {Seconds per day}
  omega_E  = 1.00273790934;  {Earth rotations per sidereal day (non-constant)}
  omega_ER = omega_E*twopi;  {Earth rotation, radians per sidereal day}

var
  ds50 : double;

Function Julian_Date_of_Year(year : double) : double;
Function Julian_Date_of_Epoch(epoch : double) : double;
Function Epoch_Time(jd : double) : double;
Function DOY(yr,mo,dy : word) : word;
Function Fraction_of_Day(hr,mi,se,hu : word) : double;
Function Calendar_Date(jd : double) : date;
Function Time_of_Day(jd : double;
                   full : boolean;
                    res : byte) : clock_time;
Function ThetaG(epoch : double) : double;
Function ThetaG_JD(jd : double) : double;
Function Delta_ET(year : double) : double;

IMPLEMENTATION
  Uses MinMax,Support;

const
  half_sec = 0.5/secday;

Function Julian_Date_of_Year(year : double) : double;
  { Astronomical Formulae for Calculators, Jean Meeus, pages 23-25 }
  { Calculate Julian Date of 0.0 Jan year }
  var
    A,B   : longint;
  begin
  year := year - 1;
  A := Trunc(year/100);
  B := 2 - A + Trunc(A/4);
  Julian_Date_of_Year := Trunc(365.25 * year)
                       + Trunc(30.6001 * 14)
                       + 1720994.5 + B;
  end; {Function Julian_Date_of_Year}

Function Julian_Date_of_Epoch(epoch : double) : double;
  var
    year,day : double;
  begin
  year := 1900 + Int(epoch*1E-3);
  day  := Frac(epoch*1E-3)*1E3;
  Julian_Date_of_Epoch := Julian_Date_of_Year(year) + day;
  end; {Function Julian_Date_of_Epoch}

Function Epoch_Time(jd : double) : double;
  var
    year,mo,dy     : word;
    yr,time,epoch : double;
    edate          : date;
  begin
  edate := Calendar_Date(jd);
  year := Integer_Value(edate,1,4);
  yr := Integer_Value(edate,3,2);
  mo := Pos(Copy(edate,6,3),'  JanFebMarAprMayJunJulAugSepOctNovDec') div 3;
  dy := Integer_Value(edate,10,2);
  time := Frac(jd + 0.5);
  Epoch_Time := yr*1000 + DOY(year,mo,dy) + time;
  end; {Function Epoch_Time}
```

```
Function DOY(yr,mo,dy : word) : word;
  const
    days : array [1..12] of word = (31,28,31,30,31,30,31,31,30,31,30,31);
  var
    i,day : word;
  begin
  day := 0;
  for i := 1 to mo-1 do
    day := day + days[i];
  day := day + dy;
  if ((yr mod 4) = 0) and
     (((yr mod 100) <> 0) or ((yr mod 400) = 0)) and
     (mo > 2) then
    day := day + 1;
  DOY := day;
  end; {Function DOY}

Function Fraction_of_Day(hr,mi,se,hu : word) : double;
  begin
  Fraction_of_Day := (hr + (mi + (se + hu/100)/60)/60)/24;
  end; {Function Fraction_of_Day}

Function Calendar_Date(jd : double) : date;
  { Astronomical Formulae for Calculators, Jean Meeus, pages 26-27 }
  var
    Z,month             : longint;
    A,B,C,D,E,F,alpha   : double;
    day,year            : double;
    syear               : string[4];
    cdate               : date;
  begin
  Z := Trunc(jd + 0.5);
  F := Frac(jd + 0.5);
  if Z < 2299161 then
    A := Z
  else
    begin
    alpha := Int((Z - 1867216.25)/36524.25);
    A := Z + 1 + alpha - Int(alpha/4);
    end; {else}
  B := A + 1524;
  C := Int((B - 122.1)/365.25);
  D := Int(365.25 * C);
  E := Int((B - D)/30.6001);
  day := B - D - Int(30.6001 * E) + F;
  if E < 13.5 then
    month := Round(E - 1)
  else
    month := Round(E - 13);
  if month > 2.5 then
    year := C - 4716
  else
    year := C - 4715;
  Str(year:4:0,syear);
  cdate := syear + ' '
         + Copy('   JanFebMarAprMayJunJulAugSepOctNovDec',3*month,3) + ' '
         + TwoDigit(Trunc(day));
  Calendar_Date := cdate;
  end; {Function Calendar_Date}

Function Time_of_Day(jd : double;
                   full : boolean;
                    res : byte) : clock_time;
  var
    hr,mn   : longint;
    time,sc : double;
    ctime   : string;
  begin
  res := IMin(IMax(0,res),3);
  time := 24 * Frac(jd - 0.5 + half_sec);
  hr   := Trunc(time);
  time := 60 * Frac(time);
  mn   := Trunc(time);
  sc   := 60 * Frac(time) - 0.5;
  time := 1000000 + 10000 * hr + 100 * mn + sc;
  Str(time:(7+res):res,ctime);
  Delete(ctime,1,1);
  if full then
    begin
    Insert(':',ctime,5);
    Insert(':',ctime,3);
    end; {if}
  Time_of_Day := ctime;
  end; {Function Time_of_Day}
```

```
Function ThetaG(epoch : double) : double;
{ Reference:  The 1992 Astronomical Almanac, page B6. }
  var
    year,day,UT,jd,TU,GMST : double;
  begin
  year := 1900 + Int(epoch*1E-3);
  day  := Frac(epoch*1E-3)*1E3;
  UT   := Frac(day);
  day  := Int(day);
  jd   := Julian_Date_of_Year(year) + day;
  TU   := (jd - 2451545.0)/36525;
  GMST := 24110.54841 + TU * (8640184.812866 + TU * (0.093104 - TU * 6.2E-6));
  GMST := Modulus(GMST + secday*omega_E*UT,secday);
  ThetaG := twopi * GMST/secday;
  ds50 := jd - 2433281.5 + UT;
{ ThetaG := Modulus(6.3003880987*ds50 + 1.72944494,twopi); }
  end; {Function ThetaG}

Function ThetaG_JD(jd : double) : double;
{ Reference:  The 1992 Astronomical Almanac, page B6. }
  var
    UT,TU,GMST : double;
  begin
  UT   := Frac(jd + 0.5);
  jd   := jd - UT;
  TU   := (jd - 2451545.0)/36525;
  GMST := 24110.54841 + TU * (8640184.812866 + TU * (0.093104 - TU * 6.2E-6));
  GMST := Modulus(GMST + secday*omega_E*UT,secday);
  ThetaG_JD := twopi * GMST/secday;
  end; {Function ThetaG_JD}

Function Delta_ET(year : double) : double;
  { Values determined using data from 1950-1991 in the 1990 Astronomical
    Almanac.  See DELTA_ET.WQ1 for details. }
  begin
  Delta_ET := 26.465 + 0.747622*(year - 1950)
             + 1.886913*Sin(twopi*(year - 1975)/33);
  end; {Function Delta_ET}

end.
```

B.6 SUPPORT Unit Source Code Listing

```
Unit Support;  (** This unit contains machine-specific code **)
{           Author:  Dr TS Kelso }
{ Original Version:  1992 Jun 25 }
{ Current Revision:  1992 Oct 20 }
{          Version:  1.90 }
{        Copyright:  1992, All Rights Reserved }
{$N+}

INTERFACE

const  {IBM PC screen codes}
  BS       =  ^H;                         {Backspace}
  CR       =  ^M;                         {Carriage Return}
  CRLF     =  ^M^J;                       {Carriage Return/Line Feed}
  BELL     =  ^G;                         {Terminal Bell}
  ESC      =  ^[;                         {Escape}
  DEL      =  #$7F;                       {Delete}
  Up       =  #72;                        {Up Cursor}
  Dn       =  #80;                        {Down Cursor}
  Rt       =  #77;                        {Right Cursor}
  Lt       =  #75;                        {Left Cursor}
  Home     =  #71;                        {Home Key}
  Endd     =  #79;                        {End Key}
  PgUp     =  #73;                        {Page Up}
  PgDn     =  #81;                        {Page Down}
  C_Lt     =  #115;                       {Control-Left Cursor}
  C_Rt     =  #116;                       {Control-Right Cursor}
  C_PgUp   =  #132;                       {Control-Page Up}
  C_PgDn   =  #118;                       {Control-Page Down}
  UpDown   =  #24#25;                     {Up/Down Arrows}
  Cursors  =  #24#25#26#27;               {Up/Down/Left/Right Arrows}
  SFrame : string = '';     {Single-Line Frame Characters}
  DFrame : string = '';     {Double-Line Frame Characters}
  MFrame : string = '';          {Mixed-Line Frame Characters}
type
  options  = array [0..10] of string;
  time_set = record
    yr,mo,dy,hr,mi,se,hu : word;
    end; {record}

Procedure Cursor_On;
Procedure Cursor_Off;
Procedure Save_Cursor;
Procedure Restore_Cursor;
Procedure ReverseVideo;
Procedure NormalVideo;
Procedure BoldVideo;
Procedure FrameWindow(x,y,w,h,color : byte; title : string);
Procedure MakeWindow(x,y,w,h,color : byte; title : string);
Procedure ClearWindow(x,y,w,h : byte);
Procedure Show_Status_Line(title : string);
Procedure Show_Instructions(title : string);
Procedure Clear_Status_Line;
Procedure Report_Error(x,y : byte; title : string);
Procedure Beep;
Procedure Buzz;
Procedure Mark_Time;
Procedure Zero_Time(var time : time_set);
Procedure Get_Current_Time(var time : time_set);
Function Yes : boolean;
Function TwoDigit(arg : integer) : string;
Function ThreeDigit(arg : integer) : string;
Procedure Convert_Blanks(var field : string);
Function Integer_Value(buffer : string;
                  start,length : integer) : integer;
Function Real_Value(buffer : string;
               start,length : integer) : double;
Function File_Exists(filename : string) : boolean;
Function Select_File(title,pattern,default : string; x,y,w,h : byte) : string;
Function Select_Option(menu : options; number,x,y,w,h : byte) : byte;

IMPLEMENTATION
  Uses CRT,DOS,MinMax;

var
  Last_X,Last_Y : byte;

Procedure Cursor_On;
  var
```

```
    regs : registers;
  begin
  with regs do
    begin
    ah := $01;
    ch := 0;
    cl := 7;
    end; {with}
  Intr($10,regs);
  end; {Procedure Cursor_On}

Procedure Cursor_Off;
  var
    regs : registers;
  begin
  with regs do
    begin
    ah := $01;
    ch := $20;
    cl := $00;
    end; {with}
  Intr($10,regs);
  end; {Procedure Cursor_Off}

Procedure Save_Cursor;
  begin
  Last_X := WhereX;
  Last_Y := WhereY;
  end; {Procedure Save_Cursor}

Procedure Restore_Cursor;
  begin
  GotoXY(Last_X,Last_Y);
  end; {Procedure Restore_Cursor}

Procedure ReverseVideo;
  begin
  TextColor(black);
  TextBackground(lightgray);
  end; {Procedure ReverseVideo}

Procedure NormalVideo;
  begin
  TextColor(lightgray);
  TextBackground(black);
  end; {Procedure NormalVideo}

Procedure BoldVideo;
  begin
  TextColor(yellow);
  TextBackground(black);
  end; {Procedure BoldVideo}

Procedure FrameWindow(x,y,w,h,color : byte;
                              title : string);
  var
    i : byte;
  begin
{  Window(x,y,x+w+3,y+h+1);}
{  ClrScr; }
  Window(x,y,x+w+3,y+h+2);
  TextColor(color);
  Write(DFrame[1]);
  for i := 1 to w+2 do
    Write(DFrame[2]);
  Write(DFrame[3]);
  for i := 1 to h do
    begin
    GotoXY(1,i+1);
    Write(DFrame[4]);
    GotoXY(w+4,i+1);
    Write(DFrame[4]);
    end; {for i}
  GotoXY(1,h+2);
  Write(DFrame[5]);
  for i := 1 to w+2 do
    Write(DFrame[2]);
  Write(DFrame[6]);
  GotoXY(2,1);
  Write(MFrame[4],Copy(title,1,w),MFrame[6]);
  NormalVideo;
  end; {Procedure FrameWindow}

Procedure MakeWindow(x,y,w,h,color : byte;
                             title : string);
```

```
  begin
  FrameWindow(x,y,w,h,color,title);
  Window(x+2,y+1,x+w+1,y+h);
  end; {Procedure MakeWindow}

Procedure ClearWindow(x,y,w,h : byte);
  begin
  Window(x,y,x+w+3,y+h+1);
  ClrScr;
  Window(1,1,80,25);
  end; {Procedure ClearWindow}

Procedure Show_Status_Line(title : string);
  begin
  GotoXY(1,25);
  Write(Copy(title,1,79));
  ClrEOL;
  end; {Procedure Show_Status_Line}

Procedure Show_Instructions(title : string);
  begin
  GotoXY(80-Length(title),25);
  Write(Copy(title,1,79));
  ClrEOL;
  end; {Procedure Show_Instructions}

Procedure Clear_Status_Line;
  begin
  Show_Status_Line('');
  end; {Procedure Clear_Status_Line}

Procedure Report_Error(x,y : byte; title : string);
  begin
  GotoXY(x,y);
  BoldVideo;
  Write(title);
  NormalVideo;
  GotoXY(1,24);
  Cursor_On;
  Halt;
  end; {Procedure Report_Error}

Procedure Beep;
  var
    i : integer;
  begin
  for i := 1 to 3 do
    begin
    Sound(1500);
    Delay(100);
    NoSound;
    Delay(10);
    end; {for}
  end; {Procedure Beep}

Procedure Buzz;
  var
    i : integer;
  begin
  for i := 1 to 3 do
    begin
    Sound(500);
    Delay(100);
    NoSound;
    Delay(10);
    end; {for}
  end; {Procedure Buzz}

Procedure Mark_Time;
  const
    time_count : byte = 0;
  begin
  case time_count of
    0 : Write('-');
    1 : Write('\');
    2 : Write('|');
    3 : Write('/');
    end; {case}
  time_count := (time_count + 1) mod 4;
  Write(^H);
  end; {Procedure Mark_Time}

Procedure Zero_Time(var time : time_set);
  begin
  with time do
```

```
    begin
    yr := 0;
    mo := 0;
    dy := 0;
    hr := 0;
    mi := 0;
    se := 0;
    hu := 0;
    end; {with}
  end; {Procedure Zero_Time}

Procedure Get_Current_Time(var time : time_set);
  var
    dw : word;
  begin
  with time do
    begin
    GetDate(yr,mo,dy,dw);
    GetTime(hr,mi,se,hu);
    end;
  end; {Procedure Get_Current_Time}

Function Yes : boolean;
  var
    ch    : char;
    valid : boolean;
  begin
  Cursor_On;
  repeat
    ch := Upcase(ReadKey);
    valid := true;
    case ch of
      'Y' : begin
            writeln('Yes');
            Yes := true;
            end; {Yes}
      'N' : begin
            writeln('No');
            Yes := false;
            end; {No}
    else
      valid := false;
    end; {case}
  until valid;
  Cursor_Off;
  end; {Function Yes}

Function TwoDigit(arg : integer) : string;
  begin
  TwoDigit := Chr((arg div 10) + Ord('0'))
            + Chr((arg mod 10) + Ord('0'));
  end; {Function TwoDigit}

Function ThreeDigit(arg : integer) : string;
  var
    hundreds,barg : integer;
  begin
  hundreds := arg div 100;
  barg := arg - 100*hundreds;
  ThreeDigit := Chr(hundreds + Ord('0')) + TwoDigit(barg);
  end; {Function ThreeDigit}

Procedure Convert_Blanks(var field : string);
  var
    i : integer;
  begin
  for i := length(field) downto 1 do
    if field[i] = ' ' then
      field[i] := '0';
  end; {Procedure Convert_Blanks}

Function Integer_Value(buffer : string;
                 start,length : integer) : integer;
  var
    answer,result : integer;
  begin
  buffer := Copy(buffer,start,length);
  Convert_Blanks(buffer);
  Val(buffer,answer,result);
  if result = 0 then
    Integer_Value := answer
  else
    Integer_Value := 0;
  end; {Function Integer_Value}

Function Real_Value(buffer : string;
              start,length : integer) : double;
  var
```

```
    minus  : byte;
    result : integer;
    answer : double;
  begin
  buffer := Copy(buffer,start,length);
  Convert_Blanks(buffer);
  if buffer = '' then
    buffer := '0';
  minus := Pos('-',buffer);
  if minus > 0 then
    begin
    buffer[minus] := '0';
    Val(buffer,answer,result);
    answer := -answer;
    end {if}
  else
    Val(buffer,answer,result);
  if result = 0 then
    Real_Value := answer
  else
    Report_Error(1,24,'Invalid call to Real_Value!');
  end; {Function Real_Value}

Function File_Exists(filename : string) : boolean;
  var
    filehandle : text;
  begin
  Assign(filehandle,filename);
  {$i-} Reset(filehandle); {$i+}
  if IOResult = 0 then
    begin
    File_Exists := true;
    Close(filehandle);
    end {if}
  else
    File_Exists := false;
  end; {Function File_Exists}

Function Select_File(title,pattern,default : string;
                                 x,y,w,h : byte) : string;
  var
    choice          : char;
    start,stop,
    count,select,i  : word;
    dirinfo         : SearchRec;
    filedata        : array [1..50] of string;
  begin
  Cursor_Off;
  FindFirst(pattern,AnyFile,dirinfo);
  count := 0;
  select := 1;
  while DosError = 0 do
    begin
    count := count + 1;
    filedata[count] := dirinfo.name;
    if filedata[count] = default then
      select := count;
    FindNext(dirinfo);
    end; {while}
  w := IMax(12,w);
  h := IMin(h,IMax(1,count));
  MakeWindow(x,y,w,h,white,title);
  if count = 0 then
    begin
    BoldVideo;
    Write('No files!');
    Delay(1000);
    NormalVideo;
    Window(1,1,80,25);
    GotoXY(1,25);
    ClrEOL;
    GotoXY(1,24);
    Cursor_On;
    Halt;
    end {if}
  else
    begin
    start := IMin(count - h + 1,select);
    stop  := start + h - 1;
    repeat
      ClrScr;
      for i := start to stop do
        begin
        GotoXY(1,i-start+1);
        if i = select then BoldVideo;
        Write(Copy(filedata[i],1,w));
        if i = select then NormalVideo;
```

```
        end; {for i}
      choice := ReadKey;
      if choice = #0 then
        begin
        choice := ReadKey;
        case choice of
          Up : begin
               select := IMax(1,select-1);
               if select < start then
                 begin
                 start := select;
                 stop  := start + h - 1;
                 end; {if}
               end; {Up}
          Dn : begin
               select := IMin(count,select+1);
               if select > stop then
                 begin
                 stop  := select;
                 start := stop - h + 1;
                 end; {if}
               end; {Dn}
          end; {case}
        end; {if}
    until choice = CR;
    Select_File := filedata[select];
    Delay(500);
    end; {else}
  MakeWindow(x,y,w,h,lightgray,title);
  Window(1,1,80,25);
  end; {Function Select_File}

Function Select_Option(menu : options;
                number,x,y,w,h : byte) : byte;
  var
    choice              : [illegible]
    start,stop,select,i : [illegible]
  begin
  Cursor_Off;
  h := IMin(h,number);
  select := 1;
  MakeWindow(x,y,w,h,white,menu[0]);
  start := IMin(number - h + 1,select);
  stop  := start + h - 1;
  repeat
    ClrScr;
    for i := start to stop do
      begin
      GotoXY(1,i-start+1);
      if i = select then BoldVideo;
      Write(menu[i]);
      if i = select then NormalVideo;
      end; {for i}
    choice := ReadKey;
    if choice = #0 then
      begin
      choice := ReadKey;
      case choice of
        Up : begin
             select := IMax(1,select-1);
             if select < start then
               begin
               start := select;
               stop  := start + h - 1;
               end; {if}
             end; {Up}
        Dn : begin
             select := IMin(number,select+1);
             if select > stop then
               begin
               stop  := select;
               start := stop - h + 1;
               end; {if}
             end; {Dn}
        end; {case}
      end; {if}
  until choice = CR;
  Select_Option := select;
  Delay(500);
  MakeWindow(x,y,w,h,lightgray,menu[0]);
  Window(1,1,80,25);
  end; {Function Select_Option}
end.
```

B.7 SGP_CONV Unit Source Code Listing

```
Unit SGP_Conv;
{           Author:  Dr TS Kelso }
{ Original Version:  1991 Oct 30}
{ Current Revision:  1992 Sep 03}
{          Version:  1.00 }
{        Copyright:  1992, All Rights Reserved }
{$N+}
INTERFACE
  Uses SGP_Math;

Procedure Convert_Satellite_Data(arg : integer);
Procedure Convert_Sat_State(var pos,vel : vector);

IMPLEMENTATION
  Uses Support,SGP_Intf,SGP_Init,SGP_Time;

Procedure Convert_Satellite_Data(arg : integer);
  var
    iexp,ibexp                 : integer;
    a1,ao,del1,delo,xnodp,temp : double;
    abuf                       : two_line;
  begin
  abuf := sat_data[arg];
{* Decode Card 1 *}
  catnr    := Copy(abuf[1],3,5);
  epoch    := Real_Value(abuf[1],19,14);
  julian_epoch := Julian_Date_of_Epoch(epoch);
  xndt2o   := Real_Value(abuf[1],34,10);
  xndd6o   := Real_Value(abuf[1],45,6)*1E-5;
  iexp     := Integer_Value(abuf[1],51,2);
  bstar    := Real_Value(abuf[1],54,6)*1E-5;
  ibexp    := Integer_Value(abuf[1],60,2);
  elset    := ThreeDigit(Integer_Value(abuf[1],66,3));
{* Decode Card 2 *}
  xincl    := Real_Value(abuf[2],9,8);
  xnodeo   := Real_Value(abuf[2],18,8);
  eo       := Real_Value(abuf[2],27,7)*1E-7;
  omegao   := Real_Value(abuf[2],35,8);
  xmo      := Real_Value(abuf[2],44,8);
  xno      := Real_Value(abuf[2],53,11);
{ period   := 1/xno; }
{* Convert to proper units *}
  xndd6o   := xndd6o*Power(10.0,iexp);
  bstar    := bstar*Power(10.0,ibexp)/ae;
  xnodeo   := Radians(xnodeo);
  omegao   := Radians(omegao);
  xmo      := Radians(xmo);
  xincl    := Radians(xincl);
  xno      := xno*twopi/xmnpda;
  xndt2o   := xndt2o*twopi/Sqr(xmnpda);
  xndd6o   := xndd6o*twopi/Cube(xmnpda);
{* Determine whether Deep-Space Model is needed *}
  a1 := Power(xke/xno,tothrd);
  temp := 1.5*ck2*(3*Sqr(Cos(xincl))-1)/Power(1 - eo*eo,1.5);
  del1 := temp/(a1*a1);
  ao := a1*(1 - del1*(0.5*tothrd + del1*(1 + 134/81*del1)));
  delo := temp/(ao*ao);
  xnodp := xno/(1 + delo);
  if (twopi/xnodp >= 225) then
    ideep := 1
  else
    ideep := 0;
  iflag := 1;
  end; {Procedure Convert_Satellite_Data}

Procedure Convert_Sat_State(var pos,vel : vector);
  var
    i : byte;
  begin
  for i := 1 to 3 do
    begin
    pos[i] := pos[i]*xkmper;      {kilometers}
    vel[i] := vel[i]*xkmper/60;  {kilometers/second}
    end; {for i}
  Magnitude(pos);
  Magnitude(vel);
  end; {Procedure Convert_Sat_State}
end.
```

B.8 SGP_IN Unit Source Code Listing

```
Unit SGP_In;  (** This unit contains machine-specific code **)
{           Author:  Dr TS Kelso }
{ Original Version:  1992 Jun 25 }
{ Current Revision:  1992 Sep 13 }
{          Version:  2.00 }
{        Copyright:  1992, All Rights Reserved }
{$N+}

INTERFACE
  Uses SGP_Math,SGP_Init,Support;

const
  data_type : byte = 3;

var
  fsat,fobs : text;

Procedure Select_Time(message : string;
                      xpos,ypos : byte;
                    var default : time_set;
                      precision : byte);
Procedure Select_Time_Interval(message : string;
                               xpos,ypos : byte;
                             var default : time_set;
                               precision : byte);
Function Checksum_Good(line : line_data) : boolean;
Function Good_Elements(line : two_line) : boolean;
Procedure Input_Satellite(index : word);
Function Input_Satellite_Data(fn : string) : word;
Procedure Select_Satellites(title : string;
                            x,y,w,h : byte;
                             number : word);
Procedure Input_Observer(var geodetic : vector);

IMPLEMENTATION
  Uses CRT,DOS,MinMax;

var
  i : byte;

Procedure Echo_Time(time : time_set;
          item,precision : byte);
  begin
  GotoXY(2,1);
  ClrEOL;
  if item = 1 then ReverseVideo;
  Write(time.yr); NormalVideo;
  Write(' ');
  if item = 2 then ReverseVideo;
  Write(Copy('   JanFebMarAprMayJunJulAugSepOctNovDec',3*time.mo,3));
  NormalVideo;
  Write(' ');
  if item = 3 then ReverseVideo;
  Write(TwoDigit(time.dy));
  NormalVideo;
  if precision > 3 then
    begin
    Write('  ');
    if item = 4 then ReverseVideo;
    Write(TwoDigit(time.hr));
    NormalVideo;
    end; {if}
  if precision > 4 then
    begin
    Write(':');
    if item = 5 then ReverseVideo;
    Write(TwoDigit(time.mi));
    NormalVideo;
    end; {if}
  if precision > 5 then
    begin
    Write(':');
    if item = 6 then ReverseVideo;
    Write(TwoDigit(time.se));
    NormalVideo;
    end; {if}
  if precision > 6 then
    begin
    Write('.');
    if item = 7 then ReverseVideo;
    Write(TwoDigit(time.hu));
    NormalVideo;
```

```
    end; {if}
  end; {Procedure Echo_Time}

Procedure Update_Field(choice : char;
                   var value : word;
                   llim,ulim : word;
                     var pos : byte;
                   lpos,upos : byte);
  begin
  case choice of
     Up : value := IMin(value+1,ulim);
     Dn : value := IMax(value-1,llim);
   Home : value := llim;
   Endd : value := ulim;
     Lt : pos := IMax(lpos,pos-1);
     Rt : pos := IMin(pos+1,upos);
    end; {case}
  end; {Procedure Update_Field}

Procedure Select_Time(message : string;
                    xpos,ypos : byte;
                  var default : time_set;
                    precision : byte);
  const
    days : array [0..1,1..12] of byte
       = ((31,28,31,30,31,30,31,31,30,31,30,31),
          (31,29,31,30,31,30,31,31,30,31,30,31));
  var
    pos,w     : byte;
    choice    : char;
    llim,ulim : array [1..7] of word;
  Function LY(year : word) : byte;
    begin
    if (year mod 4 = 0) and
       ((year mod 100 <> 0) or (year mod 400 = 0)) then
      LY := 1
    else
      LY := 0;
    end; {Function LY}
  begin
  Cursor_Off;
  precision := IMin(IMax(3,precision),7);
  case precision of
    3 : w := 13;
    4 : w := 17;
    5 : w := 20;
    6 : w := 23;
    7 : w := 26;
    end; {case}
  if precision < 7 then
    default.hu := 0;
  if precision < 6 then
    default.se := 0;
  if precision < 5 then
    default.mi := 0;
  if precision < 4 then
    default.hr := 0;
  MakeWindow(xpos,ypos,w,1,white,message);
  llim[1] := 1957;  ulim[1] := 2200;
  llim[2] :=    1;  ulim[2] :=   12;
  llim[3] :=    1;  ulim[3] := days[LY(default.yr),default.mo];
  llim[4] :=    0;  ulim[4] :=   23;
  llim[5] :=    0;  ulim[5] :=   59;
  llim[6] :=    0;  ulim[6] :=   59;
  llim[7] :=    0;  ulim[7] :=   99;
  pos := 1;
  Echo_Time(default,pos,precision);
  repeat
    choice := ReadKey;
    if choice = #00 then
      begin
      choice := ReadKey;
      if choice in [Up,Dn,Home,Endd,Lt,Rt] then
        begin
        case pos of
    1 : Update_Field(choice,default.yr,llim[pos],ulim[pos],pos,1,precision);
    2 : Update_Field(choice,default.mo,llim[pos],ulim[pos],pos,1,precision);
    3 : Update_Field(choice,default.dy,llim[pos],ulim[pos],pos,1,precision);
    4 : Update_Field(choice,default.hr,llim[pos],ulim[pos],pos,1,precision);
    5 : Update_Field(choice,default.mi,llim[pos],ulim[pos],pos,1,precision);
    6 : Update_Field(choice,default.se,llim[pos],ulim[pos],pos,1,precision);
    7 : Update_Field(choice,default.hu,llim[pos],ulim[pos],pos,1,precision);
          end; {case}
```

```
        ulim[3] := days[LY(default.yr),default.mo];
        if default.dy > ulim[3] then
          default.dy := ulim[3];
        Echo_Time(default,pos,precision);
        end; {if}
      end; {if}
  until choice = ^M;
  MakeWindow(xpos,ypos,w,1,lightgray,message);
  Echo_Time(default,0,precision);
  Window(1,1,80,25);
  end; {Procedure Input_Time}

Procedure Echo_Time_Interval(time : time_set;
                    item,precision : byte);
  begin
  GotoXY(4,2);
  ClrEOL;
  if item = 3 then ReverseVideo;
  Write(TwoDigit(time.dy));
  NormalVideo;
  if precision > 3 then
    begin
    Write('  ');
    if item = 4 then ReverseVideo;
    Write(TwoDigit(time.hr));
    NormalVideo;
    end; {if}
  if precision > 4 then
    begin
    Write(':');
    if item = 5 then ReverseVideo;
    Write(TwoDigit(time.mi));
    NormalVideo;
    end; {if}
  if precision > 5 then
    begin
    Write(':');
    if item = 6 then ReverseVideo;
    Write(TwoDigit(time.se));
    NormalVideo;
    end; {if}
  if precision > 6 then
    begin
    Write('.');
    if item = 7 then ReverseVideo;
    Write(TwoDigit(time.hu));
    NormalVideo;
    end; {if}
  end; {Procedure Echo_Time_Interval}

Procedure Select_Time_Interval(message : string;
                           xpos,ypos : byte;
                         var default : time_set;
                           precision : byte);
  var
    pos,w     : byte;
    choice    : char;
    llim,ulim : array [1..7] of word;
  begin
  Cursor_Off;
  precision := IMin(IMax(3,precision),7);
  case precision of
    3 : w :=  5;
    4 : w :=  9;
    5 : w := 12;
    6 : w := 15;
    7 : w := 18;
    end; {case}
  if precision < 7 then
    default.hu := 0;
  if precision < 6 then
    default.se := 0;
  if precision < 5 then
    default.mi := 0;
  if precision < 4 then
    default.hr := 0;
  MakeWindow(xpos,ypos,IMax(w+1,13),2,white,message);
  Write(Copy(' days  hr:mn:sc.hu',1,w));
  llim[3] := 0;  ulim[3] := 99;
  llim[4] := 0;  ulim[4] := 23;
  llim[5] := 0;  ulim[5] := 59;
  llim[6] := 0;  ulim[6] := 59;
  llim[7] := 0;  ulim[7] := 99;
  pos := 3;
```

```
Echo_Time_Interval(default,pos,precision);
repeat
  choice := ReadKey;
  if choice = #00 then
    begin
    choice := ReadKey;
    if choice in [Up,Dn,Home,Endd,Lt,Rt] then
      begin
      case pos of
        3 : Update_Field(choice,default.dy,llim[pos],ulim[pos],pos,3,precision);
        4 : Update_Field(choice,default.hr,llim[pos],ulim[pos],pos,3,precision);
        5 : Update_Field(choice,default.mi,llim[pos],ulim[pos],pos,3,precision);
        6 : Update_Field(choice,default.se,llim[pos],ulim[pos],pos,3,precision);
        7 : Update_Field(choice,default.hu,llim[pos],ulim[pos],pos,3,precision);
        end; {case}
      Echo_Time_Interval(default,pos,precision);
      end; {if}
    end; {if}
until choice = ^M;
MakeWindow(xpos,ypos,IMax(w+1,13),2,lightgray,message);
Echo_Time_Interval(default,0,precision);
Window(1,1,80,25);
end; {Procedure Input_Time_Interval}

Function Checksum_Good(line : line_data) : boolean;
  var
    i,checksum,check_digit : integer;
  begin
  checksum := 0;
  for i := 1 to 68 do
    case line[i] of
      '0'..'9' : checksum := checksum + Ord(line[i]) - Ord('0');
           '-' : checksum := checksum + 1;
      end; {case}
  checksum := checksum mod 10;
  check_digit := Ord(line[69]) - Ord('0');
  Checksum_Good := (checksum = check_digit);
  end; {Function Checksums_Good}

Function Good_Elements(line : two_line) : boolean;
  var
    result : boolean;
  begin
  result := Checksum_Good(line[1]) and Checksum_Good(line[2]);
  if (line[1,1] <> '1') or
     (line[2,1] <> '2') or
     (Copy(line[1],3,5) <> Copy(line[2],3,5)) then
    result := false;
  if (line[1,24] <> '.') or
     (line[1,35] <> '.') or
     (Copy(line[1],62,3) <> ' 0 ') or
     (line[2,12] <> '.') or
     (line[2,21] <> '.') or
     (line[2,38] <> '.') or
     (line[2,47] <> '.') or
     (line[2,55] <> '.') then
    result := false;
  Good_Elements := result;
  end; {Function Good_Elements}

Procedure Input_Satellite(index : word);
  begin
  if not EOF(fsat) then
    begin
    if data_type = 3 then
      Readln(fsat,sat_name[index]);
    Readln(fsat,sat_data[index,1]);
    Readln(fsat,sat_data[index,2]);
    end; {if}
  end; {Procedure Input_Satellite}

Function Input_Satellite_Data(fn : string) : word;
  var
    count : word;
  begin
  if data_type in [2,3] then
    begin
    count := 0;
    Assign(fsat,data_drive + data_dir + fn);
    Reset(fsat);
    repeat
      count := count + 1;
      Input_Satellite(count);
```

```
      until EOF(fsat) or (count = max_sats);
      Close(fsat);
      Input_Satellite_Data := count;
      end {if}
    else
      begin
      GotoXY(1,24);
      Writeln('Invalid data type!');
      Halt;
      end; {else}
    end; {Procedure Input_Satellite_Data}

Procedure Select_Satellites(title : string;
                            x,y,w,h : byte;
                             number : word);
  var
    choice              : char;
    start,stop,select,i : word;
  begin
  Cursor_Off;
  h := IMin(h,number);
  select := 1;
  w := IMax(w,12);
  MakeWindow(x.y,w,h,white,title);
  start := IMin(number - h + 1,select);
  stop  := start + h - 1;
  repeat
    ClrScr;
    for i := start to stop do
      begin
      GotoXY(1,i-start+1);
      if i = select then TextBackground(blue);
      if selected[i] then Write('[*] ') else Write('[ ] ');
      Write(Copy(sat_data[i,1],3,5),'  ',Copy(sat_name[i],1,w-12));
      ClrEOL;
      if i = select then TextBackground(black);
      end; {for i}
    choice := ReadKey;
    if choice = #0 then
      begin
      choice := ReadKey;
      case choice of
          Up : begin
               select := IMax(1,select-1);
               if select < start then
                 begin
                 start := select;
                 stop  := start + h - 1;
                 end; {if}
               end; {Up}
        PgUp : begin
               select := IMax(1,select-h);
               if select < start then
                 begin
                 start := select;
                 stop  := start + h - 1;
                 end; {if}
               end; {PgUp}
          Dn : begin
               select := IMin(number,select+1);
               if select > stop then
                 begin
                 stop  := select;
                 start := stop - h + 1;
                 end; {if}
               end; {Dn}
        PgDn : begin
               select := IMin(number,select+h);
               if select > stop then
                 begin
                 stop  := select;
                 start := stop - h + 1;
                 end; {if}
               end; {PgDn}
        end; {case}
      end {if}
    else
      case UpCase(choice) of
        ' ' : begin
              selected[select] := Not(selected[select]);
              select := IMin(number,select+1);
              if select > stop then
                begin
```

```
              stop  := select;
              start := stop - h + 1;
              end; {if}
            end; {Toggle}
      'A' : begin
            for i := 1 to number do
              selected[i] := Not(selected[i]);
            end; {Toggle All}
      end; {case}
  until choice = CR;
  Delay(500);
  MakeWindow(x,y,w,h,lightgray,title);
  Window(1,1,80,25);
  end; {Procedure Select_Satellites}

Procedure Input_Observer(var geodetic : vector);
  begin
  if not EOF(fobs) then
    begin
    Readln(fobs,obs_name,geodetic[1],geodetic[2],geodetic[3]);
    geodetic[1] := Radians(geodetic[1]);
    geodetic[2] := Radians(Modulus(geodetic[2],360));
    geodetic[3] := geodetic[3]*0.001;
    end; {if}
  end; {Procedure Input_Observer}

  begin
  for i := 1 to max_sats do
    selected[i] := false;

end.
```

B.9 SGP_OUT Unit Source Code Listing

```
Unit SGP_Out;
{           Author:  Dr TS Kelso }
{ Original Version:  1992 Aug 24 }
{ Current Revision:  1992 Oct 01 }
{          Version:  1.50 }
{        Copyright:  1992, All Rights Reserved }
{$N+}
INTERFACE
  Uses SGP_Math;

const
  day_date   : boolean = true;
  full_time  : boolean = true;
  N_E_W_S    : boolean = false;
  D_M_S      : boolean = false;
  time_res   : byte = 2;
  angle_res  : byte = 4;
  dist_res   : byte = 3;

var
  fout : text;

Procedure Output_Time(time : double);
Procedure Output_ECI(time : double;
                  pos,vel : vector);
Procedure Output_Angle(angle : double;
                   width,dec : byte;
                     degrees : boolean);
Procedure Output_LatLonAlt(time : double;
                       geodetic : vector);
Procedure Output_Obs(time : double;
                      obs : vector);
Procedure Output_RADec(time : double;
                        obs : vector);

IMPLEMENTATION
  Uses Support,SGP_Init,SGP_Time,Solar;

Procedure Output_Time(time : double);
  begin
  if day_date then
    begin
    Write(fout,Calendar_Date(time),'  ');
    Write(fout,Time_of_Day(time,full_time,time_res));
    end {if}
  else
    Write(fout,time:16:8);
  end; {Procedure Output_Time}

Procedure Output_ECI(time : double;
                  pos,vel : vector);
  var
    i : byte;
  begin
  Output_Time(time);
  for i := 1 to 3 do
    Write(fout,pos[i]:11:3);
  for i := 1 to 3 do
    Write(fout,vel[i]:11:6);
  if show_vis then
    if eclipsed then
      Writeln(fout,' ECL')
    else
      Writeln(fout,'    ')
  else
    Writeln(fout);
  end; {Procedure Output_ECI}

Procedure Output_Angle(angle : double;
                   width,dec : byte;
                     degrees : boolean);
  var
    asign       : shortint;
    deg,min,sec : longint;
    tmp         : double;
  begin
  if dec > 4 then
    dec := 4;
  if D_M_S then
    begin
    asign := Sign(angle);
    angle := Abs(angle);
```

```
    case dec of
      0 : begin
          angle := Modulus(angle + 0.5,360);
          deg := asign*Trunc(angle);
          end; {0}
      1 : begin
          angle := Modulus(angle + 1/12,360);
          deg := Trunc(angle);
          min := Trunc((angle - deg) * 6)*10;
          deg := asign*deg;
          end; {1}
      2 : begin
          angle := Modulus(angle + 1/120,360);
          deg := Trunc(angle);
          min := Trunc((angle - deg) * 60);
          deg := asign*deg;
          end; {2}
      3 : begin
          angle := Modulus(angle + 1/720,360);
          deg := Trunc(angle);
          tmp := (angle - deg) * 60;
          min := Trunc(tmp);
          sec := Trunc(Frac(tmp)*6)*10;
          deg := asign*deg;
          end; {3}
      4 : begin
          angle := Modulus(angle + 1/7200,360);
          deg := Trunc(angle);
          tmp := (angle - deg) * 60;
          min := Trunc(tmp);
          sec := Trunc(Frac(tmp)*60);
          deg := asign*deg;
          end; {4}
      end; {case}
    if degrees then
      case dec of
          0 : Write(fout,deg:width,'');
        1,2 : Write(fout,deg:width,'',TwoDigit(min),'''');
        3,4 : Write(fout,deg:width,'',TwoDigit(min),'''',TwoDigit(sec),'"');
        end {case}
    else
      case dec of
          0 : Write(fout,deg:width,'h');
        1,2 : Write(fout,deg:width,'h',TwoDigit(min),'m');
        3,4 : Write(fout,deg:width,'h',TwoDigit(min),'m',TwoDigit(sec),'s');
        end {case}
    end {if}
  else
    if dec = 0 then
      Write(fout,angle:width:0)
    else
      Write(fout,angle:width+dec+1:dec);
  end; {Procedure Output_Angle}

Procedure Output_LatLonAlt(time : double;
                           geodetic : vector);
  begin
  Output_Time(time);
  if N_E_W_S then
    begin
    Output_Angle(Abs(Degrees(geodetic[1])),5,angle_res,true);
    if geodetic[1] >= 0 then
      Write(fout,' N')
    else
      Write(fout,' S');
    if geodetic[2] > pi then
      geodetic[2] := geodetic[2] - twopi;
    Output_Angle(Abs(Degrees(geodetic[2])),6,angle_res,true);
    if geodetic[2] >= 0 then
      Write(fout,' E')
    else
      Write(fout,' W');
    end {if}
  else
    begin
    Output_Angle(Degrees(geodetic[1]),5,angle_res,true);
    Output_Angle(Degrees(geodetic[2]),6,angle_res,true);
    end; {if}
  Write(fout,geodetic[3]:11:3);
  if show_vis then
    if eclipsed then
```

```
      Writeln(fout,' ECL')
    else
      Writeln(fout,'    ')
  else
    Writeln(fout);
  end; {Procedure Output_LatLonAlt}

Procedure Output_Obs(time : double;
                     obs : vector);
  const
    first : boolean = false;
  begin
  if not visible then
    begin
    if first then Writeln(fout);
    first := false;
    Exit;
    end; {if}
  first := true;
  Output_Time(time);
  Write(fout,Copy(obs_name,1,3):5);
  Writeln(fout,Degrees(obs[1]):angle_res+6:angle_res,
               Degrees(obs[2]):angle_res+5:angle_res,
               obs[3]:dist_res+8:dist_res,
               obs[4]:dist_res+8:dist_res+3);
  end; {Procedure Output_Obs}

Procedure Output_RADec(time : double;
                       obs : vector);
  const
    first : boolean = false;
  begin
  if not visible then
    begin
    if first then Writeln(fout);
    first := false;
    Exit;
    end; {if}
  first := true;
  Output_Time(time);
  Write(fout,Copy(obs_name,1,3):5);
  Output_Angle(Degrees(obs[1])/15,4,angle_res,false);
  Output_Angle(Degrees(obs[2]),5,angle_res,true);
  Writeln(fout);
  end; {Procedure Output_RADec}

end.
```

B.10 SGP_OBS Unit Source Code Listing

```
Unit SGP_Obs;
{           Author:  Dr TS Kelso }
{ Original Version:  1992 Jun 02 }
{ Current Revision:  1992 Sep 28 }
{          Version:  1.40 }
{        Copyright:  1992, All Rights Reserved }
{$N+}

INTERFACE
  Uses SGP_Math;

Procedure Calculate_User_PosVel(var geodetic : vector;
                                         time : double;
                          var obs_pos,obs_vel : vector);
Procedure Calculate_LatLonAlt(pos : vector;
                             time : double;
                     var geodetic : vector);
Procedure Calculate_Obs(pos,vel,geodetic : vector;
                                    time : double;
                             var obs_set : vector);
Procedure Calculate_RADec(pos,vel,geodetic : vector;
                                      time : double;
                               var obs_set : vector);

IMPLEMENTATION
  Uses SGP_Intf,SGP_Init,SGP_Time;

Procedure Calculate_User_PosVel(var geodetic : vector;
                                         time : double;
                          var obs_pos,obs_vel : vector);
{ Reference:  The 1992 Astronomical Almanac, page K11. }
  const
    mfactor = twopi*omega_E/secday;
  var
    lat,lon,alt,
    theta,c,s,achcp : double;
  begin
  lat := geodetic[1];
  lon := geodetic[2];
  alt := geodetic[3];
  theta := Modulus(ThetaG_JD(time) + lon,twopi);
  geodetic[4] := theta; {LMST}
  c := 1/Sqrt(1 + f*(f - 2)*Sqr(Sin(lat)));
  s := Sqr(1 - f)*c;
  achcp := (xkmper*c + alt)*Cos(lat);
  obs_pos[1] := achcp*Cos(theta);            {kilometers}
  obs_pos[2] := achcp*Sin(theta);
  obs_pos[3] := (xkmper*s + alt)*Sin(lat);
  obs_vel[1] := -mfactor*obs_pos[2];         {kilometers/second}
  obs_vel[2] :=  mfactor*obs_pos[1];
  obs_vel[3] :=  0;
  Magnitude(obs_pos);
  Magnitude(obs_vel);
  end; {Procedure Calculate_User_PosVel}

Procedure Calculate_LatLonAlt(pos : vector;
                             time : double;
                     var geodetic : vector);
{ Reference:  The 1992 Astronomical Almanac, page K12. }
  var
    lat,lon,alt,
    theta,r,e2,phi,c : double;
  begin
  theta := AcTan(pos[2],pos[1]);
  lon := Modulus(theta - ThetaG_JD(time),twopi);
  r := Sqrt(Sqr(pos[1]) + Sqr(pos[2]));
  e2 := f*(2 - f);
  lat := AcTan(pos[3],r);
  repeat
    phi := lat;
    c := 1/Sqrt(1 - e2*Sqr(Sin(phi)));
    lat := AcTan(pos[3] + xkmper*c*e2*Sin(phi),r);
  until Abs(lat - phi) < 1E-10;
  alt := r/Cos(lat) - xkmper*c;
  geodetic[1] := lat;   {radians}
  geodetic[2] := lon;   {radians}
  geodetic[3] := alt;   {kilometers}
  geodetic[4] := theta; {radians}
```

```
  end; {Procedure Calculate_LatLonAlt}

Procedure Calculate_Obs(pos,vel,geodetic : vector;
                                    time : double;
                            var obs_set : vector);
  var
    i                    : integer;
    lat,lon,alt,theta,
    sin_lat,cos_lat,
    sin_theta,cos_theta : double;
    el,azim             : double;
    top_s,top_e,top_z   : double;
    obs_pos,obs_vel,
    range,rgvel         : vector;
  begin
  Calculate_User_PosVel(geodetic,time,obs_pos,obs_vel);
  for i := 1 to 3 do
    begin
    range[i] := pos[i] - obs_pos[i];
    rgvel[i] := vel[i] - obs_vel[i];
    end; {for i}
  Magnitude(range);
  lat := geodetic[1];
  lon := geodetic[2];
  alt := geodetic[3];
  theta := geodetic[4];
  sin_lat := Sin(lat);
  cos_lat := Cos(lat);
  sin_theta := Sin(theta);
  cos_theta := Cos(theta);
  top_s := sin_lat*cos_theta*range[1]
         + sin_lat*sin_theta*range[2]
         - cos_lat*range[3];
  top_e := -sin_theta*range[1]
         + cos_theta*range[2];
  top_z := cos_lat*cos_theta*range[1]
         + cos_lat*sin_theta*range[2]
         + sin_lat*range[3];
  azim := ArcTan(-top_e/top_s); {Azimuth}
  if top_s > 0 then
    azim := azim + pi;
  if azim < 0 then
    azim := azim + twopi;
  el := ArcSin(top_z/range[4]);
  obs_set[1] := azim;                          {Azimuth (radians)}
  obs_set[2] := el;                            {Elevation (radians)}
  obs_set[3] := range[4];                      {Range (kilometers)}
  obs_set[4] := Dot(range,rgvel)/range[4];     {Range Rate (kilometers/second)}
{ Corrections for atmospheric refraction }
{ Reference:  Astronomical Algorithms by Jean Meeus, pp. 101-104 }
{ Note:  Correction is meaningless when apparent elevation is below horizon }
  obs_set[2] := obs_set[2] +
Radians((1.02/Tan(Radians(Degrees(el)+10.3/(Degrees(el)+5.11))))/60);
  if obs_set[2] >= 0 then
    visible := true
  else
    begin
    obs_set[2] := el;  {Reset to true elevation}
    visible := false;
    end; {else}
  end; {Procedure Calculate_Obs}

Procedure Calculate_RADec(pos,vel,geodetic : vector;
                                     time : double;
                             var obs_set : vector);
{ Reference:  Methods of Orbit Determination by Pedro Ramon Escobal, pp. 401-402}
  var
    phi,theta,
    sin_theta,cos_theta,
    sin_phi,cos_phi,
    az,el,
    Lxh,Lyh,Lzh,
    Sx,Ex,Zx,
    Sy,Ey,Zy,
    Sz,Ez,Zz,
    Lx,Ly,Lz,
    cos_delta,
    sin_alpha,cos_alpha : double;
  begin
  Calculate_Obs(pos,vel,geodetic,time,obs_set);
  if visible then
    begin
```

```
    az := obs_set[1];
    el := obs_set[2];
    phi   := geodetic[1];
    theta := Modulus(ThetaG_JD(time) + geodetic[2],twopi);
    sin_theta := Sin(theta);
    cos_theta := Cos(theta);
    sin_phi := Sin(phi);
    cos_phi := Cos(phi);
    Lxh := -Cos(az)*Cos(el);
    Lyh :=  Sin(az)*Cos(el);
    Lzh :=  Sin(el);
    Sx := sin_phi*cos_theta;
    Ex := -sin_theta;
    Zx := cos_theta*cos_phi;
    Sy := sin_phi*sin_theta;
    Ey := cos_theta;
    Zy := sin_theta*cos_phi;
    Sz := -cos_phi;
    Ez := 0;
    Zz := sin_phi;
    Lx := Sx*Lxh + Ex*Lyh + Zx*Lzh;
    Ly := Sy*Lxh + Ey*Lyh + Zy*Lzh;
    Lz := Sz*Lxh + Ez*Lyh + Zz*Lzh;
    obs_set[2] := ArcSin(Lz);            {Declination (radians)}
    cos_delta := Sqrt(1 - Sqr(Lz));
    sin_alpha := Ly/cos_delta;
    cos_alpha := Lx/cos_delta;
    obs_set[1] := AcTan(sin_alpha,cos_alpha); {Right Ascension (radians)}
    obs_set[1] := Modulus(obs_set[1],twopi);
    end; {if}
  end; {Procedure Calculate_RADec}

end.
```

B.11 SOLAR Unit Source Code Listing

```
Unit Solar;
{           Author:  Dr TS Kelso }
{ Original Version:  1990 Jul 29 }
{ Current Revision:  1992 Sep 30 }
{          Version:  1.20 }
{        Copyright:  1990-1992, All Rights Reserved }
{$N+}

INTERFACE
  Uses SGP_Math;

const
  eclipsed : boolean = false;
  show_vis : boolean = false;

var
  civil,
  nautical,
  astronomical : double;  {Twilight elevations}
  solar_pos    : vector;

Procedure Calculate_Solar_Position(time : double;
                              var solar_vector : vector);
Function Depth_of_Eclipse(time : double;
                             r1 : vector) : double;

IMPLEMENTATION
  Uses SGP_Intf,SGP_Time;

const
  sr = 696000.0;          {Solar radius - kilometers (IAU 76)}
  AU = 1.49597870E8;      {Astronomical unit - kilometers (IAU 76)}

Procedure Calculate_Solar_Position(time : double;
                              var solar_vector : vector);
  var
    mjd,year,T,M,L,e,C,O,Lsa,nu,R,eps : double;
    ob                                : vector;
  begin
  mjd := time - 2415020.0;
  year := 1900 + mjd/365.25;
  T := (mjd + Delta_ET(year)/secday)/36525.0;
  M := Radians(Modulus(358.47583 + Modulus(35999.04975*T,360.0)
     - (0.000150 + 0.0000033*T)*Sqr(T),360.0));
  L := Radians(Modulus(279.69668 + Modulus(36000.76892*T,360.0)
     + 0.0003025*Sqr(T),360.0));
  e := 0.01675104 - (0.0000418 + 0.000000126*T)*T;
  C := Radians((1.919460 - (0.004789 + 0.000014*T)*T)*Sin(M)
     + (0.020094 - 0.000100*T)*Sin(2*M) + 0.000293*Sin(3*M));
  O := Radians(Modulus(259.18 - 1934.142*T,360.0));
  Lsa := Modulus(L + C - Radians(0.00569 - 0.00479*Sin(O)),twopi);
  nu := Modulus(M + C,twopi);
  R := 1.0000002*(1 - Sqr(e))/(1 + e*Cos(nu));
  eps := Radians(23.452294 - (0.0130125 + (0.00000164 - 0.000000503*T)*T)*T
     + 0.00256*Cos(O));
  R := AU*R;
  solar_vector[1] := R*Cos(Lsa);
  solar_vector[2] := R*Sin(Lsa)*Cos(eps);
  solar_vector[3] := R*Sin(Lsa)*Sin(eps);
  solar_vector[4] := R;
  end; {Procedure Calculate_Solar_Position}

Function Depth_of_Eclipse(time : double;
                             r1 : vector) : double;
  var
    r1_r1,r1_r2,r2_r2,k,d,ds : double;
    r2                       : vector;
  begin
  Magnitude(r1);
  Calculate_Solar_Position(time,r2);
  solar_pos := r2;
  r1_r1 := Sqr(r1[4]);
  r1_r2 := -Dot(r1,r2);
  r2_r2 := Sqr(r2[4]);
  k := r1_r2/r2_r2;
{Calculate perpendicular distance from anti-solar vector}
  d := Sqrt(r1_r1 - Sqr(r1_r2)/r2_r2);
{Calculate shadow distance ds}
  ds := xkmper + k * (sr - xkmper);
{If d < ds, then satellite is in eclipse}
  if (k > 0) and (d < ds) then
```

```
    eclipsed := true
  else
    eclipsed := false;
  Depth_of_Eclipse := d - ds
  end; {Function Depth_of_Eclipse}

  begin
  civil        := Radians(-6);
  nautical     := Radians(-12);
  astronomical := Radians(-18);

end.
```

B.12 MINMAX Unit Source Code Listing

```
Unit MinMax;
{            Author:  Dr TS Kelso }
{ Original Version:  1992 Jun 29 }
{ Current Revision:  1992 Sep 03 }
{           Version:  1.02 }
{         Copyright:  1992, All Rights Reserved }
{$N+}

INTERFACE

Function IMin(arg1,arg2 : integer) : integer;
Function IMax(arg1,arg2 : integer) : integer;
Function RMin(arg1,arg2 : real) : real;
Function RMax(arg1,arg2 : real) : real;
Function DMin(arg1,arg2 : double) : double;
Function DMax(arg1,arg2 : double) : double;

IMPLEMENTATION

Function IMin(arg1,arg2 : integer) : integer;
  begin
  if arg1 < arg2 then
    IMin := arg1
  else
    IMin := arg2;
  end; {Function IMin}

Function IMax(arg1,arg2 : integer) : integer;
  begin
  if arg1 > arg2 then
    IMax := arg1
  else
    IMax := arg2;
  end; {Function IMax}

Function RMin(arg1,arg2 : real) : real;
  begin
  if arg1 < arg2 then
    RMin := arg1
  else
    RMin := arg2;
  end; {Function RMin}

Function RMax(arg1,arg2 : real) : real;
  begin
  if arg1 > arg2 then
    RMax := arg1
  else
    RMax := arg2;
  end; {Function RMax}

Function DMin(arg1,arg2 : double) : double;
  begin
  if arg1 < arg2 then
    DMin := arg1
  else
    DMin := arg2;
  end; {Function DMin}

Function DMax(arg1,arg2 : double) : double;
  begin
  if arg1 > arg2 then
    DMax := arg1
  else
    DMax := arg2;
  end; {Function DMax}

end.
```

Appendix C. *TRUTH MODEL*

C.1 HPOP_IN Source Code Listing

```
Program HPOP_Input;
{                Author:  Dr TS Kelso }
{    Original Version:  1992 Aug 20 }
{    Current Revision:  1992 Oct 14 }
{               Version:  2.25 }
{ Program Information:  Convert two-line element sets to state vectors
                        for input to HPOP/SAM. }
{$N+}
  Uses CRT,Support,
  SGP_Init,SGP_In,
  SGP_Conv,
  SGP_Math,SGP_Time,
  SGP4SDP4;

var
  timei,times,timee : time_set;
  jei,je2           : double;
  sat_pos,sat_vel   : vector;
  fni,fno           : string;
  fi                : text;

Function HPOP_Time(tm : time_set) : string;
  begin
  with tm do
    HPOP_Time := TwoDigit(yr div 100)
               + TwoDigit(yr mod 100)
               + TwoDigit(mo)
               + TwoDigit(dy)
               + TwoDigit(hr)
               + TwoDigit(mi)
               + TwoDigit(se) + '.'
               + TwoDigit(hu);
  end; {Function HPOP_Time}

Function HPOP_ETime(je : double) : string;
  var
    time  : double;
    edate : date;
  begin
  edate := Calendar_Date(je);
  with timee do
    begin
    yr := Integer_Value(edate,1,4);
    mo := Pos(Copy(edate,6,3),' JanFebMarAprMayJunJulAugSepOctNovDec') div 3;
    dy := Integer_Value(edate,10,2);
    time := 24*Frac(je + 0.5);
    hr := Trunc(time);
    time := 60*(time - hr);
    mi := Trunc(time);
    time := 60*(time - mi);
    se := Trunc(time);
    hu := Trunc(100*Frac(time));
    end; {with}
  HPOP_ETime := HPOP_Time(timee);
  end; {Procedure Convert_Time}

Procedure Output_HPOP;
  var
    i   : byte;
    int : double;
    fo  : text;
  begin
  Assign(fo,work_drive+work_dir+fno);
  Rewrite(fo);
  for i := 1 to 3 do
    Writeln(fo,1000*sat_pos[i]:12:3);
  for i := 1 to 3 do
    Writeln(fo,1000*sat_vel[i]:15:6);
  with times do
    int := hu/100 + se + (mi + (hr + (24*dy))*60)*60;
  Writeln(fo,int:12:3);
  Writeln(fo,HPOP_ETime(jei));
  with timei do
    je2 := jei + dy + Fraction_of_Day(hr,mi,se,hu);
  Writeln(fo,HPOP_ETime(jei));
  Writeln(fo,HPOP_ETime(je2));
  Writeln(fo,'0.01');
```

```
  Close(fo);
  end; {Procedure Output_HPOP}

BEGIN

  Program_Initialize('HPOP-IN');
  Show_Status_Line('Select input two-line elements');
  Show_Instructions('<'+UpDown+' to select, ENTER when done>');
  fni := Select_File('Two-Line Elements',data_drive+data_dir+'*.2LE','',40,2,19,5);
  Assign(fi,data_drive+data_dir+fni);
  Reset(fi);
  Zero_Time(timei);
  Zero_Time(times);
  timei.dy := 60;
  times.mi := 5;
  Show_Status_Line('Time interval from epoch');
  Show_Instructions('<'+cursors+'/Home/End to select, ENTER when done>');
  Select_Time_Interval('Time Interval',40,9,timei,7);
  Clear_Status_Line;
  Show_Status_Line('Output time step');
  Show_Instructions('<'+cursors+'/Home/End to select, ENTER when done>');
  Select_Time_Interval('Time Step',40,13,times,7);
  Clear_Status_Line;
  GotoXY(1,24);
  repeat
    Mark_Time;
    Readln(fi,sat_data[1,1]);
    Readln(fi,sat_data[1,2]);
    fno := 'SV-' + Copy(sat_data[1,1],3,5)
         + '.' + ThreeDigit(Integer_Value(sat_data[1,1],66,3));
    Convert_Satellite_Data(1);
    je1 := Julian_Date_of_Epoch(epoch);
    SGP(je1,sat_pos,sat_vel);
    Convert_Sat_State(sat_pos,sat_vel);
    Output_HPOP;
  until EOF(fi);
  Close(fi);

END.
```

C.2 HPOP_IN Output/HPOP Input File Format

ECI Position(x)	Initial Position(meters)
ECI Position(y)	
ECI Position(z)	
ECI Velocity(x)	Initial Velocity (meters/second)
ECI Velocity(y)	
ECI Velocity(z)	
Step Size	Propagation step size (seconds)
Epoch Time	Epoch time of the position (YYYYMMDDHHMMSS.SSS)
Start Time	Propagation start time (YYYYMMDDHHMMSS.SSS)
Stop Time	Propagation stop time (YYYYMMDDHHMMSS.SSS)
Area-Mass Ratio	Area mass ration of the satellite (m^2/kg)

C.3 Sample HPOP_IN Output/HPOP Input File

File: SV-19859.147

```
-7575194.314
-1791719.890
       9.481
     -75.291742
    -618.161792
    7868.435726
     300.000
19900302075911.43
19900302075911.43
19900502075911.43
0.01
```

C.4 Sample HPOP Output/CONVERT Input File

FILE: PV-19859.147

```
      Time          Position(x)   Position(y)     Position(z)  Velocity(x) Velocity(y) Velocity(z)
19900302075911.43 -7575194.31400 -1791719.89000       9.48100   -75.29174  -618.16179  7868.43572
19900302080411.43 -7314412.61195 -1907912.85517  2331213.45559  1787.88725  -155.40460  7578.49848
19900302080911.43 -6527544.75443 -1887036.66329  4496446.34278  3403.31635   287.14653  6784.48618
19900302081411.43 -5309821.98319 -1741845.94718  6367459.24499  4647.97494   668.23752  5646.17234
19900302081911.43 -3779138.83937 -1494310.15383  7866660.24678  5490.94581   967.76235  4331.69393
19900302082411.43 -2052321.28041 -1169644.27783  8962430.39496  5964.13216  1183.05842  2975.18718
        .
        .
        .
        .
        .
19900502073411.43 -8749947.01735  -800119.68735  5088236.46983  1904.16474  -346.79907  5884.16506
19900502073911.43 -8037873.88455  -890440.06104  6760184.84992  2813.11029  -255.42774 5244.07122
19900502074411.43 -7079394.34900  -953554.65123  8225888.43876  3549.03791  -165.80210 4515.64102
19900502074911.43 -5924349.23870  -990274.92476  9464050.80267  4125.98788   -79.68859 3731.70528
19900502075411.43 -4618259.12963 -1001824.01191 10461703.70189  4558.31158     1.87721 2915.46436
19900502075911.43 -3202524.58393  -989663.55330 11211748.20752  4858.93881    78.30139 2083.25883
```

time:	in YYYYMMDDHHMMSS.SSS Format
position:	in meters
velocity:	in meters per second

C.5 CONVERT Program Source Code Listing

```
Program Convert;
{                 Author:  Dr TS Kelso }
{    Original Version:  1992 Jul 22 }
{    Current Revision:  1992 Oct 16 }
{             Version:  2.03 }
{ Program Information:  Convert state vectors to observations. }
{$N+}
  Uses CRT,Support,
       SGP_Init,SGP_In,
       SGP_Math,SGP_Time,
       SGP_Obs,Solar;

const
  max_sites = 100;

var
  first,
  limits,dark     : boolean;
  i,nr_sites      : byte;
  yr,mo,dy,
  hr,mi,se,hu     : word;
  time,DoE        : double;
  obs_pos,obs_vel,
  obs_set,
  solar_obs,
  sat_pos,sat_vel : vector;
  min_range,
  max_range,
  min_el,
  max_el,
  az1,az2         : array [1..max_sites] of double;
  site            : array [1..max_sites] of vector;
  site_des        : array [1..max_sites] of string[3];
  h1,h2           : string[2];
  tm              : string[18];
  name,buffer     : string[25];
  fn1,fn2,fn3,fn4 : string;
  fi2,fo,fi4      : text;

Procedure Convert_Time(tm : string;
 var yr,mo,dy,hr,mi,se,hu : word);
  begin
  yr := Integer_Value(tm,1,4);
  mo := Integer_Value(tm,5,2);
  dy := Integer_Value(tm,7,2);
  hr := Integer_Value(tm,9,2);
  mi := Integer_Value(tm,11,2);
  se := Integer_Value(tm,13,2);
  hu := Integer_Value(tm,16,2);
  end; {Procedure Convert_Time}

Procedure Output_Data(arg : byte;
                     time : double;
           obs_set,satpos : vector);
  begin
  Writeln(fo,site_des[arg],'  ',
             time:16:8,'  ',
             Degrees(obs_set[1]):7:3,'  ',
             Degrees(obs_set[2]):7:3,'  ',
             obs_set[3]:9:3,'  ',
             obs_set[4]:9:3,'  ',
             sat_pos[1]:10:3,'  ',
             sat_pos[2]:10:3,'  ',
             sat_pos[3]:10:3);
  end; {Procedure Output_Data}

Function Check_Limits(obs : vector;
                      arg : byte) : boolean;
  var
    good : boolean;
  begin
  good := visible and
```

```
          (obs[3] >= min_range[arg]) and
          (obs[3] <= max_range[arg]);
  if good then
    begin
    obs[1] := Degrees(obs[1]);
    obs[2] := Degrees(obs[2]);
    if (az2[arg] > az1[arg]) then
      good := (obs[1] >= az1[arg]) and
              (obs[1] <= az2[arg]) and
              (obs[2] >= min_el[arg]) and
              (obs[2] <= max_el[arg])
    else
      good := (((obs[1] >= az1[arg]) and (obs[1] <= 360)) or
               ((obs[1] >= 0) and (obs[1] <= az2[arg]))) and
              (obs[2] >= min_el[arg]) and
              (obs[2] <= max_el[arg]);
    if not good and (max_el[arg] > 90) then
      begin
      if obs[1] > 180 then
        obs[1] := obs[1] - 180
      else
        obs[1] := obs[1] + 180;
      obs[2] := 180 - obs[2];
      if (az2[arg] > az1[arg]) then
        good := (obs[1] >= az1[arg]) and
                (obs[1] <= az2[arg]) and
                (obs[2] <= max_el[arg])
      else
        good := (((obs[1] >= az1[arg]) and (obs[1] <= 360)) or
                 ((obs[1] >= 0) and (obs[1] <= az2[arg]))) and
                (obs[2] <= max_el[arg]);
      end; {if}
    end; {if}
  Check_Limits := good;
  end; {Function Check_Limits}

BEGIN

  Program_Initialize('CONVERT');
  if ParamCount >= 1 then
    fn2 := ParamStr(1)
  else
    fn2 := Select_File('Input SVs',data_drive+data_dir+'PV-*.*','',40,1,20,5);
  Assign(fi2,data_drive+data_dir+fn2);
  Reset(fi2);
  if ParamCount = 2 then
    fn1 := ParamStr(2)
  else
    fn1 := Select_File('Observation Sites',data_drive+data_dir+'*.OBS','',40,8,20,5);
  fn4 := Copy(fn1,1,Pos('.',fn1)) + 'LIM';
  if File_Exists(data_drive+data_dir+fn4) then
    begin
    Assign(fi4,data_drive+data_dir+fn4);
    Reset(fi4);
    Readln(fi4);
    limits := true;
    end {if}
  else
    limits := false;
  GotoXY(1,24);
  Cursor_Off;
  Assign(fobs,data_drive+data_dir+fn1);
  Reset(fobs);
  fn3 := fn2;
  fn3[1] := 'O';
  fn3[2] := 'B';
  i := 0;
  repeat
    i := i + 1;
    Input_Observer(site[i]);
    site_des[i] := Copy(obs_name,1,3);
    if limits then
      Readln(fi4,buffer,min_range[i],max_range[i],
                        min_el[i],max_el[i],az1[i],az2[i]);
```

```
  until EOF(fobs);
  Close(fobs);
  if limits then
    Close(fi4);
  nr_sites := i;
  Assign(fo,work_drive+work_dir+fn3);
  Rewrite(fo);
  repeat
    Readln(fi2,buffer);
  until Copy(buffer,1,4) = 'Time';
  GotoXY(1,24);
  Show_Status_Line('Processing '+fn2+' with '+fn1+' ');
  first := true;
  repeat
    Mark_Time;
    Readln(fi2,tm,sat_pos[1],sat_pos[2],sat_pos[3],
                  sat_vel[1],sat_vel[2],sat_vel[3]);
    Convert_Time(tm,yr,mo,dy,hr,mi,se,hu);
    time := Julian_Date_of_Year(yr) + DOY(yr,mo,dy)
          + Fraction_of_Day(hr,mi,se,hu);
    if first then
      begin
      Writeln(fo,'     ',time:16:8,'                                  ',
              '                                              ');
      first := false;
      end; {if}
    for i := 1 to 3 do
      begin
      sat_pos[i] := sat_pos[i] * 0.001;
      sat_vel[i] := sat_vel[i] * 0.001;
      end; {for i}
    DoE := Depth_of_Eclipse(time,sat_pos);
    for i := 1 to nr_sites do
      begin
      Calculate_Obs(sat_pos,sat_vel,site[i],time,obs_set);
      if visible and limits then
        visible := Check_Limits(obs_set,i);
      if visible then
        if Pos(site_des[i],'027 201 202 206 207 211 212 213 221 222 223 '
                         + '231 232 233 241 242 243 951 952') > 0 then
          begin
          Calculate_Obs(solar_pos,zero,site[i],time,solar_obs);
          dark := (solar_obs[2] < civil);
          visible := dark and not eclipsed;
          if visible then
            Calculate_RADec(sat_pos,sat_vel,site[i],time,obs_set);
          end; {if}
      if visible then
        Output_Data(i,time,obs_set,sat_pos);
      end; {for i}
  until EOF(fi2);
  Close(fo);
  Close(fi2);
  Program_End;
END.
```

C.6 RSELECT Program Source Code Listing

```
Program RSelect;
{               Author:  Dr TS Kelso }
{    Original Version:  1992 Sep 15 }
{    Current Revision:  1992 Oct 22 }
{              Version:  2.50 }
{ Program Information:  Randomly select observations and apply noise. }
{$N+}
  Uses CRT,Support,
       SGP_Init,
       Gauss2;

type
  data = record
    line : array [1..98] of char;
    crlf : array [1..2] of char;
    end; {record}

var
  used               : array [0..1000] of boolean;
  opd,
  i,j,l,m            : byte;
  run                : char;
  k,kk               : word;
  nobs,day,nruns : word;
  count,daily,selection : longint;
  time,last          : double;
  sd                 : array [0..100,1..4] of double;
  index              : array [0..100] of longint;
  buffer             : data;
  names,
  fn2                : string;
  fi2                : file of data;
  fno                : array [1..10] of string;
  fo                 : array [1..10] of text;

Procedure Output_Data(obs_nr : byte;
                       odata : data);
  var
    i,site    : byte;
    ob,grn    : array [1..4] of double;
    time,spos : string;
  begin
  for i := 1 to 4 do
    grn[i] := GRandom(i);
  with odata do
    begin
    if Pos(Copy(line,1,3),names) <> 0 then
      site := (Pos(Copy(line,1,3),names) - 1) div 4
    else
      begin
      for i := 1 to (nobs div 2) do
        Close(fo[i]);
      Close(fi2);
      Report_Error(1,24,'Site not found in SENSORS.OBS!');
      end; {else}
    time := Copy(line,6,16);
    ob[1] := Real_Value(line,25,7) + grn[1]*sd[site,1];
    ob[2] := Real_Value(line,34,7) + grn[2]*sd[site,2];
    if Abs(ob[2]) > 90 then
      if ob[2] < 0 then
        begin
        ob[2] := -(180 + ob[2]);
        ob[1] := ob[1] + 180;
        end {if}
      else
        begin
        ob[2] := 180 - ob[2];
        ob[1] := ob[1] + 180;
        end; {else}
    if ob[1] >= 360 then
      ob[1] := ob[1] - 360;
    ob[3] := Real_Value(line,43,9) + grn[3]*sd[site,3];
```

```
    ob[4] := Real_Value(line,54,9) + grn[4]*sd[site,4];
    spos := Copy(line,63,36);
    for i := obs_nr to nobs do
      if (i mod 2) = 0 then
        Writeln(fo[i div 2],Copy(line,1,5),time,'   ',
                ob[1]:7:3,'  ',
                ob[2]:7:3,'  ',
                ob[3]:9:3,'  ',
                ob[4]:9:3,spos);
    end; {with}
  end; {Procedure Output_Data}

Procedure Input_Covariances;
  var
    name : string[25];
    fi : text;
  begin
  Assign(fi,data_drive+data_dir+'SENSORS.COV');
  Reset(fi);
  names := '';
  i := 0;
  repeat
    Readln(fi,name,sd[i,1],sd[i,2],sd[i,3],sd[i,4]);
    names := names + Copy(name,1,4);
    i := i + 1;
  until EOF(fi);
  Close(fi);
  end; {Procedure Input_Covariances}

BEGIN

{ Randomize; }
  Program_Initialize('RSELECT4');
  if ParamCount = 0 then
    Report_Error(41,1,'Observations per day needed.')
  else
    nobs := Integer_Value(ParamStr(1),1,2);
  if ParamCount >= 2 then
    nruns := Integer_Value(ParamStr(2),1,2)
  else
    nruns := 1;
  if ParamCount = 3 then
    fn2 := ParamStr(3)
  else
    fn2 := Select_File('Input OB file',work_drive+work_dir+'OB-*.*','',40,1,20,5);
  Assign(fi2,work_drive+work_dir+fn2);
  Reset(fi2);
  GotoXY(1,24);
  Cursor_Off;
  Show_Status_Line('Indexing '+fn2+'... ');
  last := 0;
  count := 0;
  day := 0;
  repeat
    Mark_Time;
    Read(fi2,buffer);
    time := Real_Value(buffer.line,6,16);
    if time > last+1 then
      begin
      if last = 0 then
        last := time
      else
        last := last + 1;
      index[day] := count;
      day := day + 1;
      end; {if}
    count := count + 1;
  until EOF(fi2);
  index[0] := 1;
  index[day] := count;
  for opd := 1 to (nobs div 2) do
    fno[opd] := 'RO-'+ Copy(fn2,4,Pos('.',fn2)-3) + TwoDigit(2*opd);
  run := '@';
  Input_Covariances;
  for l := 1 to nruns do
    begin
```

```
    run := Succ(run);
    Show_Status_Line('Writing '+Copy(fno[1],1,8)+'/'+run+'... ');
    for opd := 1 to (nobs div 2) do
      begin
      Assign(fo[opd],work_drive+work_dir+fno[opd]+run);
      Rewrite(fo[opd]);
      end; {for}
    for i := 1 to day do
      begin
      daily := index[i] - index[i-1];
      if nobs > daily then
        Report_Error(41,1,'Not enough daily observations.');
      for k := 0 to 1000 do
        used[k] := false;
      for j := 1 to nobs do
        begin
        Mark_Time;
        selection := Random(daily - j + 1);
        kk := 0;
        for k := 0 to selection do
          while used[k+kk] do
            kk := kk + 1;
        used[k+kk] := true;
        Seek(fi2,index[i-1] + k + kk);
        Read(fi2,buffer);
        Output_Data(j,buffer);
        end; {for j}
      end; {for i}
    for opd := 1 to (nobs div 2) do
      Close(fo[opd]);
  end; {for l}
  Close(fi2);
  Program_End;

END.
```

C.7 GAUSS2 Unit Source Code Listing

```
Unit Gauss2;
{                 Author:  Dr TS Kelso }
{     Original Version:  1992 Sep 25 }
{     Current Revision:  1992 Oct 15 }
{              Version:  2.00 }
{ Program Information:  Generates gaussian random deviates from ten
                        separate streams. }
{$N+}

INTERFACE

const
  seed   : array [0..9] of longint = (428956419,
                                      1954324947,
                                      1145661099,
                                      1835732737,
                                       794161987,
                                      1329531353,
                                       200496737,
                                       633816299,
                                      1410143363,
                                      1282538739);

Function GRandom(stream : byte) : double;

IMPLEMENTATION

Function GRandom(stream : byte) : double;
{ Reference:  Numerical Recipes, page ? }
  const
    first : array [0..9] of boolean = (true,true,true,true,true,
                                       true,true,true,true,true);
    gset  : array [0..9] of double = (0,0,0,0,0,0,0,0,0,0);
  var
    v1,v2,r,fac : double;
  begin
  if first[stream] then
    begin
    RandSeed := seed[stream];
    repeat
      v1 := 2*Random - 1;
      v2 := 2*Random - 1;
      r := Sqr(v1) + Sqr(v2);
    until r < 1;
    seed[stream] := RandSeed;
    fac := Sqrt(-2*Ln(r)/r);
    gset[stream] := v1*fac;
    GRandom := v2*fac;
    first[stream] := false;
    end {if}
  else
    begin
    GRandom := gset[stream];
    first[stream] := true;
    end; {else}
  end; {Function GRandom}

end.
```

C.8 Sample RSELECT Output/DIFC Input File

FILE: RO-19859.02A.

399	2447952.83277118	180.260	4.990	3954.273	-6.293	-7575.194	-1791.720	0.009
399	2447953.23554896	128.886	11.981	8953.068	1.532	11677.050	2790.835	-573.010
394	2447954.11749340	102.560	45.089	5398.992	3.137	4879.586	393.597	9707.166
394	2447954.66610451	122.979	6.551	4909.489	-4.309	-5593.156	-1755.160	6098.198
394	2447955.13485451	355.709	67.227	3817.028	0.282	618.988	-616.404	9946.998
394	2447955.69388[illegible]29	129.836	21.914	4032.044	-3.440	-4791.167	-1629.560	7086.593
.								
.								
.								
.								
.								
399	2448011.14527118	179.474	12.840	5090.680	2.924	9656.396	632.935	-1203.649
932	2448011.36402118	257.270	69.927	4562.461	-0.777	10157.474	297.073	3585.215
344	2448012.34318785	73.006	54.620	5267.152	0.887	-6145.328	-1006.698	9170.930
386	2448012.56541007	88.497	62.601	4455.353	0.964	-7896.458	-923.961	6898.255
932	2448013.31193785	161.727	16.582	5490.990	2.815	9352.451	594.601	-1743.675
382	2448013.70082674	247.128	73.356	3805.563	0.879	-8839.462	-784.460	4778.215

Where the column headings, from right to left, are: Sensor Number, Julian Date, Azimuth or Right Ascension (degrees), Flevation or Declination (degrees), Range (kilometers), Range Rate (km/sec), ECI Position (X) (km), ECI Position (Y) (km), and ECI Position (Z) (km).

Appendix D. *DIFFERENTIAL CORRECTOR PROGRAM*

D.1 DIFC Program Source Code Listing

```
Program DIFC;
{$E+,N+}
{           Author:  BMW          }
{ Original Version:  22 Oct 1992 }
{ Current Revision:  30 Oct 1992 }
{          Version:  2.00         }

{ Description:  Sequentially differentially corrects the classical elements }

Uses  DC_Init,              {constants, argument types, variables}
      SGP_Conv,             {procedure Convert_Satellite_Data}
      SGP_Init SGP_Intf,    {constant, argument types, varialbles}
      SGP_Math,             {function Radians}
      SGP_Time,             {function Epoch_Time}
      LOWB,                 {procedure Load_Sensor_Info}
      Prop,                 {procedure Propagate}
      DC_Out,               {procedures Init_Output_Files & Close_Output_Files}
      DC_Calc,              {differential correction calculations}
      Support,              {Integer_Value, Initialize}
      Crt;

BEGIN
  Program_Initialize('DIFC');
  IF ParamCount <> 3 THEN
    begin
      writeln('INPUT FORMAT ERROR!');
      writeln('Command Line Format:  DIFC RO-satnr.nn? aa bb');
      writeln('               where:  RO-satnr.nn? is the random observation file,');
      writeln('                       aa is the correction interval in days, and');
      writeln('                       bb is the number of correction batches desired.');
      Halt;
    end; {IF}
  n_batches  := Integer_Value(ParamStr(3),1,2);
  LUPI       := Integer_Value(ParamStr(2),1,2);
  OPD        := Integer_Value(copy(ParamStr(1),10,2),1,2);
  batch_size := OPD * LUPI;

{Input sensor location & covariances}
  Load_Sensor_Data(Geodetic_SSN,Qinv_SSN);
{Input a single two line element set - the estimated elset}
  Input_Sat_OE_data(ParamStr(1));
{Input Satellite OE Set as a string and seperate the elements}
  Convert_Satellite_data(1);
{Input Satellite OE Set as a string and seperate the elements}
  Orig_Julian_epoch := Julian_epoch;
{Save epoch of seed OE set for use in calc VMAGT}
  Init_Output_Files(ParamStr(1),ParamStr(2));
{Init output files for VMAGT & Residuals}
  Input_COVinv(ParamStr(1),est_COV_inv);
{Input the 7x7 COV_inv matrix - the estimated COV}
  Invert(est_COV_inv,est_COV);
{Init est_elset array elements to OE set elements
  Init_est_elset(est_elset);
  Output_elset('Original Elset',est_elset);
  Assign(OBSinfile,work_drive+work_dir+ParamStr(1));
  Reset(OBSinfile);
  FOR batch_num := 1 TO n_batches DO
    Begin
      writeln(' ');

    {Loop to input a batch of truth observations, & calc VMAGT}

      for i := 1 to batch_size do
        begin
          Readln(OBSinfile,sensor_num,obtime_set[i],
                 true_obs_set[i,1],true_obs_set[i,2],true_obs_set[i,3],true_obs_set[i,4],
                 true_pos_set[i,1],true_pos_set[i,2],true_pos_set[i,3]);
          true_obs_set[i,1] := Radians(true_obs_set[i,1]);
          true_obs_set[i,2] := Radians(true_obs_set[i,2]);
          site_set[i] := Pos(sensor_num,sitestring) div 4;

     {Set observation type.  If optical site ensure range (obset[3]) & range rate }
     {(obset[4]) = 0.0  If NAVSPASUR or EGLIN, ensure range rate (obset[4]) = 0.0 }
```

```
          IF POS(sensor_num,'211 221 231 241 951 952 ') = 0
            THEN
              obtype_set[i] := 3
            ELSE
              begin
                obtype_set[i] := 5;
                true_obs_set[i,3] := 0.0;
                true_obs_set[i,4] := 0.0;
              end; {ELSE}
          IF POS(sensor_num,'745 398 399 ') <> 0 THEN
            begin
              obtype_set[i] := 2;
              true_obs_set[i,4] := 0.0;
            end; {THEN}

        {Calc & Output VMAGT, flag if first pass VMAGT > 30 km}

          Calc_VMAGT(obtime_set[i],true_pos_set[i],est_elset,VMAGT);
          Writeln(FVoutfile,obtime_set[i]-Orig_Julian_epoch:5:2,' ',VMAGT:7:2);
          IF (VMAGT > 30.0) AND (Batch_num > 1) THEN
            writeln(Routfile,'VMAGT = ',VMAGT:8:2,' km for Batch ',batch_num,' Ob ',i);
        end; {for i}

      {Propagate the est elset & est COV to new epoch (time of node crossing prior to last ob}

        prop_time := obtime_set[batch_size] - est_elset[0];
        Propagate(prop_time,est_elset,est_COV,prop_est_elset,prop_est_COV);
        Invert(prop_est_COV,prop_est_COV_inv);

      {Bayes Filter Algorithm}

        bstar_flag := false;
        Correct(true_obs_set,obtime_set,site_set,obtype_set,
                prop_est_elset,prop_est_COV_inv,cor_elset,cor_COV);

     {Calc & Output VMAGT after convergence}

        for i := 1 to batch_size do
          begin
            Calc_VMAGT(obtime_set[i],true_pos_set[i],cor_elset,VMAGT);
            Writeln(LVoutfile,obtime_set[i]-Orig_Julian_epoch:5:2,' ',VMAGT:7:2);
          end; {for i}

     {Declare corrected elset & COV to be est elset & COV}

        for i := 0 to 7 do
          est_elset[i] := cor_elset[i];
        for i := 1 to 7 do
          for j := 1 to 7 do
            est_COV[i,j] := cor_COV[i,j];

      end;  {FOR Batch_num}

    Close(OBSinfile);

  {Output Final corrected elset & COV for comparison to propagated original elset}

    Output_elset('Final Corrected Elset',est_elset);
    Output_COV('Final COV',est_COV);

  {Propagate the original elset to time of final corrected elset for comparisons & Output}

    Input_Sat_OE_data(ParamStr(1));              {Re-input original OE set data}
    Convert_Satellite_data(1);
    Init_est_elset(est_elset);
    prop_time := obtime_set[batch_size] - est_elset[0];
    Propagate(prop_time,est_elset,est_COV,prop_est_elset,prop_est_COV);
    Output_elset('Propagated Original Elset',prop_est_elset);

    Close_Output_Files;

  END.
```

D.2 DC_INIT Unit Source Code Listing

```
{$E+,N+}
UNIT DC_Init;

{          Author:  BMW         }
{ Original Version:  21 Oct 1992 }
{ Current Revision:  30 Oct 1992 }
{         Version:  2.00        }

{ Description:  Contains the constants, type declarations, and variables }
{               needed in for Units DC_Calc & DC_Out and Program DIFC.   }

INTERFACE
  Uses SGP_Math;               {argument type vector}

const
  del_bstar  = 1.0E-05;        {Order of Magnitude of the Median delta values of elements}
  del_xincl  = 1.0E-07;        {based on satellite catalog as of JAN 1986}
  del_xnodeo = 1.0E-06;
  del_eo     = 1.0E-08;
  del_omegao = 1.0E-06;
  del_xmo    = 1.0E-06;
  del_xno    = 1.0E-10;

  max_sites  = 30;
  max_batch_size = 100;

type
  matrix7x4       = array [1..7,1..4] of double;
  matrix4x7       = array [1..4,1..7] of double;
  matrix7x7       = array [1..7,1..7] of double;
  matrix8x1       = array [0..7] of double;
  matrixBx1       = array [1..max_batch_size] of double;
  matrixBx4       = array [1..max_batch_size] of vector;
  matrixBx1_byte  = array [1..max_batch_size] of byte;
  matrixMx4       = array [1..max_sites] of vector;

var
  i,j,k           : byte;           {counters}
  batch_num,                        {sequential correction counter in DIFC}
  obnum           : byte;           {observation number index for Bayes}
  sensor_num      : string[4];      {sensor number}
  Geodetic_SSN,                     {array for sensor geodetic Lat, Lon, & Alt}
  Sigmas_SSN,                       {array for sensor sigmas}
  Qinv_SSN        : matrixMx4;      {array for sensor covariances}
  true_obs_set    : matrixBx4;      {array for true observations}
  true_pos_set    : matrixBx4;      {array for true positions}
  obtime_set      : matrixBx1;      {array for observation times}
  site_set        : matrixBx1_byte; {array for cross referenced sensor number}
  obtype_set      : matrixBx1_byte; {array for sensor obtype (2,3, or 5)}
  cor_COV,                          {corrected ref covariance}
  prop_est_COV,                     {propagated est covariance}
  prop_est_COV_inv,                 {propagated est covariance inverse}
  est_COV,                          {est covariance}
  est_COV_inv    : matrix7x7;       {est covariance inverse}
  cor_elset,                        {corrected reference elset}
  est_elset,                        {estimated elset}
  prop_est_elset : matrix8x1;       {propagated est elset}
  Orig_Julian_epoch,               {Julian epoch of seed 2LE}
  prop_time      : double;         {propagation time in minutes}
  OBSinfile,                       {True obseravation input file ID}
  Routfile,                        {Flags output file ID}
  LVoutfile,                       {Last pass VMAGT output file ID}
  FVoutfile       : text;          {First pass VMAGT output file ID}
  n_batches,                       {number of correction batches}
  OPD,                             {Observation rate (obs/day)}
  LUPI,                            {correction update interval}
  batch_size     : integer;        {number of obs in a correction batch}
  bstar_flag,                      {True if bstar is not used for Bayes correction}
  converged      : boolean;        {Used in convergence test}
  iteration_num  : byte;           {Bayes iteration counter}
  converge_flag  : byte;           {flag for last Bayes correction run}
  VMAGT          : double;         {vector magnitude of diff in true & calc pos}
  description    : string;         {Elset output header variable}
  sitestring     : string;

IMPLEMENTATION

END.
```

D.3 DC_CALC Unit Source Code Listing

```
{$E+,N+}
UNIT DC_Calc;

{          Author:  BMW            }
{ Original Version:  23 Oct 1992 }
{ Current Revision:  30 Oct 1992 }
{          Version:  2.00          }

{ Description:  Contains the procedures used to perform the calculations }
{               for Program DIFC.                                        }

INTERFACE
  Uses DC_Init,             {constants, argument types, variables}
       SGP_Math;            {argument type vector}

Procedure Input_Sat_OE_Data(fin1 : string);
Procedure Init_est_elset(var est_elset: matrix8x1);
Procedure Calc_VMAGT(obtime : double;
                   true_pos : vector;
                      elset : matrix8x1;
                   var VMAGT : double);
Procedure Correct(true_obs_set : matrix8x4;
                    obtime_set : matrix8x1;
          site_set, obtype_set : matrix8x1_byte;
                 old_est_elset : matrix8x1;
                   old_COV_inv : matrix7x7;
             var new_est_elset : matrix8x1;
                   var new_COV : matrix7x7);
Procedure Input_COVinv(fin2 : string;
                 var COVinv : matrix7x7);
Procedure Init_7x7Array(var A : matrix7x7);
Procedure Init_7x1Array(var A : matrix8x1);
Procedure Form_T_Matrix(ref_elset : matrix8x1;
                   obtime, obtype : double;
                         Geodetic : vector;
                     var calc_obs : vector;
                            var T : matrix4x7);
Procedure Calc_Residual(true_obs, calc_obs : vector;
                              var Residual : vector);
Procedure Sum_TQT_TQR(T : matrix4x7;
        Qinv, Residual : vector;
       var Sum_TtQinvT : matrix7x7;
       var Sum_TtQinvR : matrix8x1);
Procedure Invert(M : matrix7x7;
          var Minv : matrix7x7);
Procedure Calc_new_COV(old_COV_inv, Sum_TtQinvT : matrix7x7;
                   var new_COV_inv, new_COV : matrix7x7);
Procedure Calc_corrected_ref_elset(old_COV_inv, Sum_TtQinvT, new_COV : matrix7x7;
                                        old_est_elset, ref_elset : matrix8x1;
                        var delta_ref_elset, corrected_ref_elset : matrix8x1);
Procedure Test_Convergence(delta_ref_elset : matrix8x1;
                               new_COV_inv : matrix7x7;
                         var converge_flag : byte;
                             var converged : boolean);

IMPLEMENTATION
Uses SGP_Init,SGP_Intf,       {constants, argument types, variables}
     DC_Out,                  {output procedures}
     SGP4SDP4,                {SGP propagation}
     SGP_Time,                {function Epoch_Time}
     SGP_Obs,                 {procedures Calculate_Obs & Calculate RADec}
     SGP_Conv,                {procedure Convert_Sat_State}
     Support;                 {procedure Mark_time}

var
  T             : matrix4x7;     {observation matrix (partial obs w.r.t ref_elset)}
  Sum_TtQinvT   : matrix7x7;     {running sum}
  Sum_TtQinvR   : matrix8x1;     {running sum}
  Residual,                      {True_obs - calc_obs}
  calc_pos,                      {calculated position}
  calc_vel,                      {calculated velocity}
  calc_obs       : vector;       {calculated observations}

Procedure Input_Sat_OE_Data(fin1 : string);
  var
    OEinfile      : text;
  begin
    Assign(OEinfile,data_drive+data_dir+'2LE'+copy(fin1,4,5)+'.dat');
    Reset(OEinfile);
    Readln(OEinfile,sat_data[1,1]);
    Readln(OEinfile,sat_data[1,2]);
```

```
    Close(OEinfile);
  end;  {Procedure Input_Sat_Data}

Procedure Init_est_elset(var est_elset: matrix8x1);
  begin
    est_elset[0] := Julian_epoch;
    est_elset[1] := bstar;
    est_elset[2] := xincl;
    est_elset[3] := xnodeo;
    est_elset[4] := eo;
    est_elset[5] := omegao;
    est_elset[6] := xmo;
    est_elset[7] := xno;
  end; {Procedure Init_est_elset}

Procedure Input_COVinv(fin2 : string;
                   var COVinv : matrix7x7);
  var
    COVinfile      : text;
  begin
    Assign(COVinfile,data_drive+data_dir+'COV'+copy(fin2,4,5)+'.dat');
    Reset(COVinfile);
    for i := 1 to 7 do
      Readln(COVinfile,COVinv[i,1],COVinv[i,2],COVinv[i,3],COVinv[i,4],
                       COVinv[i,5],COVinv[i,6],COVinv[i,7]);
    Close(COVinfile);
  end;  {Procedure Input_old_COV_inv}

Procedure Init_7x7Array(var A : matrix7x7);
  begin
    for i := 1 to 7 do
      for j := 1 to 7 do
        A[i,j] := 0.0;
  end; {Procedure Init_7x7Array}

Procedure Init_7x1Array(var A : matrix8x1);
  begin
    for i := 0 to 7 do
      A[i] := 0.0;
  end; {Procedure Init_7x1Array}

Procedure Calc_VMAGT(obtime : double;
                   true_pos : vector;
                      elset : matrix8x1;
                 var VMAGT : double);
  begin
    iflag := 1;                       {SGP4SDP4 flag necessary to calc 70 constants}
    Julian_epoch := elset[0];
    epoch        := Epoch_Time(elset[0]);
    bstar        := elset[1];         {Insures elements are re-initialized to the proper}
    xincl        := elset[2];         {value prior to reading observations 2 to "n"}
    xnodeo       := elset[3];
    eo           := elset[4];
    omegao       := elset[5];
    xmo          := elset[6];
    xno          := elset[7];

  {Calculate calc_obs based on SGP propagation}

    SGP(obtime,calc_pos,calc_vel);
    Convert_Sat_State(calc_pos,calc_vel);
    VMAGT := sqrt(sqr(true_pos[1]-calc_pos[1]) +
                  sqr(true_pos[2]-calc_pos[2]) +
                  sqr(true_pos[3]-calc_pos[3]));
  end; {Procedure Calc_VMAGT}

Procedure Correct(true_obs_set : matrix8x4;
                    obtime_set : matrix8x1;
         site_set, obtype_set : matrix8x1_byte;
                 old_est_elset : matrix8x1;
                   old_COV_inv : matrix7x7;
             var new_est_elset : matrix8x1;
                   var new_COV : matrix7x7);
  var
    ref_elset,
    corrected_ref_elset,
    delta_ref_elset   : matrix8x1;
    new_COV_inv       : matrix7x7;
  label
    10;
  begin
    converge_flag := 0;
    iteration_num := 0;
    for i := 0 to 7 do
      ref_elset[i] := old_est_elset[i];
10: Init_7x7Array(Sum_TtQinvT);         {Init Sum_TtQinvT array elements to 0.0}
```

```
    Init_7x1Array(Sum_TtQinvR);          {Init Sum_TtQinvR array elements to 0.0}
    Init_7x1Array(delta_ref_elset);      {Init delta_ref_elset array elements to 0.0}

  {Iteration counter & Screen print of Bayes Correction status}

    iteration_num := iteration_num + 1;
    writeln(paramstr(1),' batch ',batch_num,' iteration ',
                         iteration_num,' converge ',converge_flag);

  {Beginning of basic Bayes Correction loop}

    for obnum := 1 to batch_size do
      begin
        Mark_Time;
        Form_T_Matrix(ref_elset,obtime_set[obnum],obtype_set[obnum],
                      Geodetic_SSN[site_set[obnum]],calc_obs,T);
        Calc_Residual(true_obs_set[obnum], calc_obs, Residual);
    {   Output_Residuals(true_obs_set[obnum],calc_obs, residual);     }
        Sum_TQT_TQR(T, Qinv_SSN[site_set[obnum]], Residual, Sum_TtQinvT, Sum_TtQinvR);
      end; {for obnum}
    Calc_new_COV(old_COV_inv,Sum_TtQinvT,new_COV_inv,new_COV);
    Calc_corrected_ref_elset(old_COV_inv,Sum_TtQinvT,new_COV,old_est_elset,
                             ref_elset,delta_ref_elset,corrected_ref_elset);
    Test_Convergence(delta_ref_elset, new_COV_inv, converge_flag, converged);
    IF NOT (converged)
      THEN
        begin
          for i := 0 to 7 do
            ref_elset[i] := corrected_ref_elset[i];
          GOTO 10;
        end
      ELSE
        for i := 0 to 7 do
          new_est_elset[i] := corrected_ref_elset[i];
  end; {Procedure Correct}

Procedure Form_T_Matrix(ref_elset : matrix8x1;
                     obtime, obtype : double;
                           Geodetic : vector;
                       var calc_obs : vector;
                              var T : matrix4x7);
  var
    del_pos,
    del_vel,
    del_obs     : vector;     {calc_obs changed by small change in ref_elset}
    del_elset   : matrix8x1;  {small change to elements (del_xndt2o, del_xndd6O, etc.)}
    del         : matrix8x1;  {binary counter for change in ref_elset}
  begin
    iflag := 1;                              {SGP4SDP4 flag necessary to calc 70 constants}
    Julian_epoch := ref_elset[0];
    epoch        := Epoch_Time(ref_elset[0]);
    bstar        := ref_elset[1];            {Insures elements are re-initialized to the proper}
    xincl        := ref_elset[2];            {value prior to reading observations 2 to "n"}
    xnodeo       := ref_elset[3];
    eo           := ref_elset[4];
    omegao       := ref_elset[5];
    xmo          := ref_elset[6];
    xno          := ref_elset[7];

  {Calculate calc_obs based on SGP propagation}

      SGP(obtime,calc_pos,calc_vel);
      Convert_Sat_State(calc_pos, calc_vel); {pos from DU to km, vel from DU/min to km/sec}
      IF obtype <> 5
        THEN
          Calculate_Obs(calc_pos, calc_vel, Geodetic, obtime, calc_obs)
        ELSE
          begin
            Calculate_RADec(calc_pos, calc_vel, Geodetic, obtime, calc_obs);
            calc_obs[3] := 0.0;
            calc_obs[4] := 0.0;
          end; {else}
       IF obtype = 2 THEN
         calc_obs[4] := 0.0;

  {Assigns deltas for the T Matrix element calculation process}

      for i := 1 to 7 do
        del[i] := 0.0;
      del_elset[1] := del_bstar;
      del_elset[2] := del_xincl;
      del_elset[3] := del_xnodeo;
      del_elset[4] := del_eo;
      del_elset[5] := del_omegao;
      del_elset[6] := del_xmo;
```

```
      del_elset[7] := del_xno;

  {Loop that computes the T Matrix elements}

    for i:= 1 to 7 do
      begin
        del[i] := 1.0;
        bstar  := ref_elset[1] + del_bstar  * del[1];
        xincl  := ref_elset[2] + del_xincl  * del[2];
        xnodeo := ref_elset[3] + del_xnodeo * del[3];
        eo     := ref_elset[4] + del_eo     * del[4];
        omegao := ref_elset[5] + del_omegao * del[5];
        xmo    := ref_elset[6] + del_xmo    * del[6];
        xno    := ref_elset[7] + del_xno    * del[7];
        del[i] := 0.0;

        iflag := 1;                 {SGP4SDP4 flag necessary to calc 70 constants}

      {Calculate del_obs using varied elements}

        SGP(obtime,del_pos,del_vel);
        Convert_Sat_State(del_pos, del_vel); {pos from DU to km, vel from DU/min to km/sec}
        IF obtype <> 5
          THEN
            Calculate_Obs(del_pos, del_vel, Geodetic, obtime, del_obs)
          ELSE
            begin
              Calculate_RADec(del_pos, del_vel, Geodetic, obtime, del_obs);
              del_obs[3] := 0.0;
              del_obs[4] := 0.0;
            end; {else}
         IF obtype = 2 THEN
           del_obs[4] := 0.0;

      {Calculate Observation Matrix elements}

        for j := 1 to 4 do
          begin
            T[j,i] := (del_obs[j] - calc_obs[j]) / del_elset[i];
          end; {for j}
      end; { for i}

      IF (bstar_flag) THEN
        for i:= 1 to 4 do
          T[i,1] := 0.0;

  end; {Procedure Form_T_Matrix}

Procedure Calc_Residual(true_obs, calc_obs : vector;
                              var Residual : vector);
  begin
    for i := 1 to 4 do
      Residual[i] := true_obs[i] - calc_obs[i];
    if Residual[1] > pi then
      Residual[1] := Residual[1] - twopi;
    if Residual[1] < -1.0*pi then
      Residual[1] := Residual[1] + twopi;
  end; {Procedure Calc_Residual}

Procedure Sum_TQT_TQR(T : matrix4x7;
          Qinv, Residual : vector;
        var Sum_TtQinvT : matrix7x7;
        var Sum_TtQinvR : matrix8x1);
  var
    TtQinv  : matrix7x4;
    TtQinvT : matrix7x7;
    TtQinvR : matrix8x1;
  begin
    for i := 1 to 7 do
      for j := 1 to 4 do
        TtQinv[i,j] := T[j,i]*Qinv[j];
    for i:= 1 to 7 do
      begin
        TtQinvR[i] := TtQinv[i,1]*Residual[1] + TtQinv[i,2]*Residual[2]
                    + TtQinv[i,3]*Residual[3] + TtQinv[i,4]*Residual[4];
        for j := 1 to 7 do
          TtQinvT[i,j] := TtQinv[i,1]*T[1,j] + TtQinv[i,2]*T[2,j]
                        + TtQinv[i,3]*T[3,j] + TtQinv[i,4]*T[4,j];
      end; {for i}
    for i:= 1 to 7 do
      begin
        Sum_TtQinvR[i] := Sum_TtQinvR[i] + TtQinvR[i];
        for j := 1 to 7 do
          Sum_TtQinvT[i,j] := Sum_TtQinvT[i,j] + TtQinvT[i,j];
      end; {for i}
  end; {Procedure Sum_TQT_TQR}
```

```
Procedure Invert(M : matrix7x7;
          var Minv : matrix7x7);
  label
    abort;
  const
    attributes = 7;
    nest       = 7;
  var
    i,j,k,l,irow,icol,ll            : integer;
    determ,pivot,hold,sum,t,ab,big : double;
    index                           : array[1..nest,1..3] of integer;

  Procedure Swap(var a,b : double);
    var
      hold : double;
    begin
      hold := a;
      a := b;
      b := hold;
    end; {Procedure Swap}

{Gauss-Jordan inversion}
  begin
    for i := 1 to attributes do
      index[i,3] := 0;
    determ := 1;
    for i := 1 to attributes do
      begin {Search for largest element}
      big := 0;
      for j := 1 to attributes do
        begin
          IF index[j,3] <> 1 THEN
            begin
              for k := 1 to attributes do
                begin
                  IF index[k,3] > 1 THEN
                    begin
                      writeln('ERROR:  Matrix singular');
                      goto abort;
                    end; {IF}
                  IF index[k,3] < 1 THEN
                  IF Abs(M[j,k]) > big THEN
                    begin
                      irow := j;
                      icol := k;
                      big := Abs(M[j,k]);
                    end; {IF}
                end; {for k}
            end; {IF}
        end; {for j}
      index[icol,3] := index[icol,3] + 1;
      index[i,1] := irow;
      index[i,2] := icol;
  {Interchange rows to put pivot on diagonal}
      IF irow <> icol THEN
        begin
          determ := -determ;
          for l := 1 to attributes do
            Swap(M[irow,l],M[icol,l]);
        end; {IF irow <> icol}
  {Divide pivot row by pivot column}
      pivot := M[icol,icol];
      determ := determ * pivot;
      M[icol,icol] := 1;
      for l := 1 to attributes do
        M[icol,l] := M[icol,l] / pivot;
  {Reduce nonpivot rows}
      for ll := 1 to attributes do
        begin
          IF ll <> icol THEN
            begin
              t := M[ll,icol];
              M[ll,icol] := 0;
              for l := 1 to attributes do
                M[ll,l] := M[ll,l] - M[icol,l] * t;
            end; {IF ll <> icol}
        end; {for ll}
    end; {for i}
{Interchange columns}
    for i := 1 to attributes do
      begin
      l := attributes - i + 1;
      IF index[l,1] <> index[l,2] THEN
        begin
```

```
        irow := index[1,1];
        icol := index[1,2];
        for k := 1 to attributes do
          swap(M[k,irow],M[k,icol]);
      end; {IF index}
    end; {for i}
  for k := 1 to attributes do
    IF index[k,3] <> 1 THEN
      begin
        writeln('ERROR:  Matrix singular');
        goto abort;
      end; {IF index}
  for i := 1 to attributes do
    for j := 1 to attributes do
      Minv[i,j] := M[i,j];
  abort:
end; {Procedure Invert}

Procedure Calc_new_COV(old_COV_inv, Sum_TtQinvT : matrix7x7;
                   var new_COV_inv, new_COV : matrix7x7);
  begin
    for i := 1 to 7 do
      for j := 1 to 7 do
        new_COV_inv[i,j] := old_COV_inv[i,j] + Sum_TtQinvT[i,j];
    Invert(new_COV_inv,new_COV)
  end; {Procedure Calc_new_COV}

Procedure Calc_corrected_ref_elset(old_COV_inv, Sum_TtQinvT, new_COV : matrix7x7;
                                        old_est_elset, ref_elset : matrix8x1;
                          var delta_ref_elset, corrected_ref_elset : matrix8x1);
  var
    temp     : matrix8x1;
    max_eo,                    {max eccentricity based on mean motion}
    max_abs_bstar,             {Absolute max value of bstar based on analysis of sat data}
    max_xno  : double;         {max mean motion based on eccentricity}
  begin
    for i := 1 to 7 do
      temp[i] := old_COV_inv[i,1] * (old_est_elset[1] - ref_elset[1])
               + old_COV_inv[i,2] * (old_est_elset[2] - ref_elset[2])
               + old_COV_inv[i,3] * (old_est_elset[3] - ref_elset[3])
               + old_COV_inv[i,4] * (old_est_elset[4] - ref_elset[4])
               + old_COV_inv[i,5] * (old_est_elset[5] - ref_elset[5])
               + old_COV_inv[i,6] * (old_est_elset[6] - ref_elset[6])
               + old_COV_inv[i,7] * (old_est_elset[7] - ref_elset[7])
               + Sum_TtQinvR[i];
    for i := 1 to 7 do
      delta_ref_elset[i] := new_COV[i,1] * temp[1]
                          + new_COV[i,2] * temp[2]
                          + new_COV[i,3] * temp[3]
                          + new_COV[i,4] * temp[4]
                          + new_COV[i,5] * temp[5]
                          + new_COV[i,6] * temp[6]
                          + new_COV[i,7] * temp[7];

    corrected_ref_elset[0] := ref_elset[0];
    for i := 1 to 7 do
      corrected_ref_elset[i] := ref_elset[i] + delta_ref_elset[i];

  {Test for corrected elements being set to values outside allowable range}

    max_abs_bstar  := 1.0;
    IF (Abs(corrected_ref_elset[1]) > max_abs_bstar)  THEN
      begin
        writeln(Routfile,'Corrected bstar out of limits.  Retained ref bstar.');
        writeln(Routfile,'  Batch  ',batch_num,'  Iteration ',iteration_num);
        writeln(Routfile,'  ref bstar = ',ref_elset[1]:15);
        writeln(Routfile,'  cor bstar = ',corrected_ref_elset[1]:15);
        corrected_ref_elset[1] := ref_elset[1];
        delta_ref_elset[1] := 0.0;
        bstar_flag := true;
      end; {IF}

    max_eo  := 1.0 - Power(1.0/xke,2/3) * Power(corrected_ref_elset[7],2/3);
    IF (corrected_ref_elset[4] > max_eo) OR (corrected_ref_elset[4] < 0.0) THEN
      begin
        writeln(Routfile,'Corrected eo out of limits.  Retained ref eo.');
        writeln(Routfile,'  Batch  ',batch_num,'  Iteration ',iteration_num);
        writeln(Routfile,'  ref eo = ',ref_elset[4]:15);
        writeln(Routfile,'  cor eo = ',corrected_ref_elset[4]:15);
        writeln(Routfile,'     xno = ',corrected_ref_elset[7]:15);
        corrected_ref_elset[4] := ref_elset[4];
      end; {IF}

    max_xno := Power((1.0-corrected_ref_elset[4]),3/2)*xke;
    IF (corrected_ref_elset[7] > max_xno) OR (corrected_ref_elset[7] < 0.0) THEN
```

```
      begin
        writeln(Routfile,'Corrected xno out of limits. Retained ref xno.');
        writeln(Routfile,' Batch  ',batch_num,' Iteration ',iteration_num);
        writeln(Routfile,' ref xno = ',ref_elset[4]:15);
        writeln(Routfile,' cor xno = ',corrected_ref_elset[4]:15);
        writeln(Routfile,'      eo = ',corrected_ref_elset[7]:15);
        corrected_ref_elset[7] := ref_elset[7];
      end; {IF}
  end; {Procedure Calc_corrected_ref_elset}

Procedure Test_Convergence(delta_ref_elset : matrix8x1;
                              new_COV_inv : matrix7x7;
                        var converge_flag : byte;
                            var converged : boolean);
  var
    Sum_Squares,
    Weighted_RMS          : double;
    temp                  : matrix8x1;
  begin
    Sum_Squares := 0.0;
    for i := 1 to 7 do
      begin
        temp[i] := 0.0;
        for j := 1 to 7 do
          temp[i] := temp[i] + delta_ref_elset[j]*new_COV_inv[j,i];
      end; {for i}
    for i := 1 to 7 do
      Sum_Squares := Sum_Squares + temp[i]*delta_ref_elset[i];
    Weighted_RMS := sqrt(Sum_Squares/7);

  {When converged, & last pass data gathered, return boolean to allow exit}

    IF ((Weighted_RMS < 0.01) AND (converge_flag = 1)) OR (iteration_num = 50)
      THEN
        converged := true
      ELSE
        converged := false;

  {Output Flag if Bayes Loop did not converge, ie. iterations = 50}

    IF iteration_num = 50 THEN
      writeln(Routfile,'Bayes loop did not converge for batch ',batch_num);

  {When converged, set converge_flag and run once more for "last pass" data}

    IF (Weighted_RMS < 0.01) THEN
      converge_flag := 1;
  end;  {Procedure Test_Convergence}

END.
```

```
{$E+,N+}
UNIT DC_Out;

{            Author:  BMW          }
{ Original Version:  22 Oct 1992 }
{ Current Revision:  30 Oct 1992 }
{           Version:  2.00         }

{ Description:  Contains the procedures used to output data for Unit }
{               DC_Calc and Program DIFC.                            }

INTERFACE
Uses DC_Init,              {argument types}
     SGP_Math;             {argument type vector, function degrees}

Procedure Init_Output_Files(fin1, fin2 : string);
Procedure Close_Output_Files;
Procedure Output_elset(description : string;
                       elset : matrix8x1);
Procedure Output_Residuals(true_obs : vector;
                   calc_obs, residual : vector);
Procedure Output_COV(description: string;
                     COV : matrix7x7);

IMPLEMENTATION
Uses SGP_Init,             {argument type sat_data}
     SGP_Time;             {Function Epoch_Time}

Procedure Init_Output_Files(fin1, fin2 : string);
  begin
    Assign(Routfile,work_drive+work_dir+'R'+copy(fin1,4,5)+
                copy(fin1,10,2)+'.'+fin2+copy(fin1,12,1));
    Rewrite(Routfile);
    Assign(FVoutfile,work_drive+work_dir+'F'+copy(fin1,4,5)+
                copy(fin1,10,2)+'.'+fin2+copy(fin1,12,1));
    Rewrite(FVoutfile);
    Assign(LVoutfile,work_drive+work_dir+'L'+copy(fin1,4,5)+
                copy(fin1,10,2)+'.'+fin2+copy(fin1,12,1));
    Rewrite(LVoutfile);
  end; {Procedure Init_Output_Files}

Procedure Close_Output_Files;
  begin
    close(Routfile);
    close(FVoutfile);
    close(LVoutfile);
  end; {Procedure Close_Output_Files}

Procedure Output_elset(description : string;
                       elset : matrix8x1);
  begin
    writeln(Routfile,'********   ',description,'   ********');
    writeln(Routfile,' ');
    writeln(Routfile,'epoch              = ',Epoch_Time(elset[0]):16:8);
    writeln(Routfile,'Julian_epoch       = ',elset[0]:16:8);
    writeln(Routfile,'elset[1] (bstar)   = ',elset[1]:15,'          = ',elset[1]:11:7);
    writeln(Routfile,'elset[2] (xincl)   = ',elset[2]:15,' Rad      = ',
                     degrees(elset[2]):8:4,'  Deg');
    writeln(Routfile,'elset[3] (xnodeo)  = ',elset[3]:15,' Rad      = ',
                     degrees(elset[3]):8:4,'  Deg');
    writeln(Routfile,'elset[4] (eo)      = ',elset[4]:15,'          = ',elset[4]:11:7);
    writeln(Routfile,'elset[5] (omegao)  = ',elset[5]:15,' Rad      = ',
                     degrees(elset[5]):8:4,'  Deg');
    writeln(Routfile,'elset[6] (xmo)     = ',elset[6]:15,' Rad      = ',
                     degrees(elset[6]):8:4,'  Deg');
    writeln(Routfile,'elset[7] (xno)     = ',elset[7]:15,' Rad/min = ',
                     24*60/twopi*elset[7]:12:8,'  Rev/day');
    writeln(Routfile,' ');
  end; {Procedure Output_Elset_Info}

Procedure Output_Residuals(true_obs, calc_obs, residual : vector);
  begin
    writeln(Routfile,'****  Residuals for Batch ',batch_num,', Ob ',obnum,'  ****');
    writeln(Routfile,' ');
    writeln(Routfile,'True Obs: ',true_obs[1]:9:6,' ',true_obs[2]:9:6,' ',
                     true_obs[3]:10:4,' ',true_obs[4]:9:6);
    writeln(Routfile,'Calc Obs: ',calc_obs[1]:9:6,' ',calc_obs[2]:9:6,' ',
                     calc_obs[3]:10:4,' ',calc_obs[4]:9:6);
    writeln(Routfile,'Residual: ',residual[1]:9:6,' ',residual[2]:9:6,' ',
```

```
                                   residual[3]:10:4,' ',residual[4]:9:6);
    writeln(Routfile,'  ');
  end; {Procedure Output_Residuals}

Procedure Output_COV(description: string;
                              COV : matrix7x7);
  begin
    writeln(Routfile,'********   ',description,'   ********'),
    writeln(Routfile,'  ');
    for i := 1 to 7 do
      writeln(Routfile,COV[i,1]:15,' ',COV[i,2]:15,' ',COV[i,3]:15,' ',
              COV[i,4]:15,' ',COV[i,5]:15,' ',COV[i,6]:15,' ',COV[i,7]:15);
    writeln(Routfile,' ');
  end; {Procedure Output_new_COV}

END.
```

D.5 LOWB Unit Source Code Listing

```
{$E+,N+}
Unit LOWB;

{           Author:  BNW          }
{ Original Version:  22 Oct 1992 }
{ Current Revision:  30 Oct 1992 }
{          Version:  2.00         }

{ Description:  Contains the procedures used to read sensor Lat, Lon, and      }
{               Alt from the file SENSORS.OBS into a 2-D Array (Geodetic_SSN),}
{               and reads sensor sigma values from the file SENSORS.COV and    }
{               calculates each sensors covariance matrix (Qinv_SSN).          }
{               The arrays are indexed by a consecutive number (1-30) which    }
{               is cross referenced to the sensor ID number using the Pascal  }
{               POS function and a string of all sensor ID numbers identified }
{               as POSstring.                    }

INTERFACE
Uses DC_Init;                  {argument type matrixMx4}

Procedure Load_Sensor_Data(var Geodetic_SSN, Qinv_SSN : matrixMx4);

IMPLEMENTATION
Uses  SGP_Init,                 {Procedure Program_Initialize}
      SGP_Math;                 {function Radians}

var
  OBSfile,              {Var for the *.OBS file containing sensor pos data}
  COVfile       : text; {Var for the *.COV file containing sensor cov data}

Procedure Load_Sensor_Data(var Geodetic_SSN, Qinv_SSN : matrixMx4);
  var
    name                          : string[23];
    sensornum                     : string[4];
    i,j,site                      : byte;
    sigma1,sigma2,sigma3,sigma4   : double;
    latitude,longitude,altitude   : double;

  begin
    for i := 1 to max_sites do     {Init Geodetic_SSN & Qinv_SSN arrays}
      for j := 1 to 4 do
        begin
          Geodetic_SSN[i,j] := 0.0;
          Qinv_SSN[i,j]     := 0.0;
        end; {for j}

 { Values for Geodetic Latitude, Longitude, and Altitude of each sensor}
 { Source:  AFSPACECOM SATTRACK Program, updated by SSC OAN              }
 {                                                                       }
 {        Notation:  Latitude:   North is +    South is -                }
 {                   Longitude:  Always expressed as + East              }
 {                   Altitude:   Expressed in km AMSL                    }

    assign(OBSfile,data_drive+data_dir+'SENSORS.OBS');
    reset(OBSfile);
    sitestring := '    ';
    while not EOF (OBSfile) do
      begin
        Readln(OBSfile,sensornum,name,latitude,longitude,altitude);
        sitestring := sitestring + Copy(sensornum,1,4);
        site := Pos(sensornum,sitestring) div 4;
        Geodetic_SSN[site,1] := Radians(latitude);     {convert deg to radians}
        Geodetic_SSN[site,2] := Radians(longitude);    {convert deg to radians}
        Geodetic_SSN[site,3] := altitude * 0.001;      {convert m to km}
      end; {while not eof}
    close(OBSfile);

 {Inverse Sensor Covariance (Qinv_SSN) Matrix for all sensors.}

    assign(COVfile,data_drive+data_dir+'SENSORS.COV');
    reset(COVfile);
    while not EOF (COVfile) do
      begin
        Readln(COVfile,sensornum,name,sigma1,sigma2,sigma3,sigma4);
        site := Pos(sensornum,sitestring) div 4;
        IF sigma1 = 0.0
          THEN                              {inverse of Azimuth or Right}
            Qinv_SSN[site,1] := 0.0         {Ascension sigma (Radians) squared}
          ELSE
            Qinv_SSN[site,1] := 1/sqr(radians(sigma1));
        IF sigma2 = 0.0
```

```
        THEN                                    {inverse of Elevation or Declenation}
          Qinv_SSN[site,2] := 0.0               {sigma (Radians) squared}
        ELSE
          Qinv_SSN[site,2] := 1/sqr(radians(sigma2));
      IF sigma3 = 0.0
        THEN
          Qinv_SSN[site,3] := 0.0               {inverse of Range sigma (km) squared or zero}
        ELSE
          Qinv_SSN[site,3] := 1/sqr(sigma3);
      IF sigma4 = 0.0
        THEN
          Qinv_SSN[site,4] := 0.0               {inverse of Range Rate (km/sec) sigma squared or zero}
        ELSE
          Qinv_SSN[site,4] := 1/sqr(sigma4);
    end; {while not eof}
  close(COVfile);
 end; {Procedure Load_Sensor_Data}
END.
```

D.6 PROP Unit Source Code Listing

```
{$E+,N+}
Unit Prop;

{           Author:  BMW          }
{ Original Version:  21 Oct 1992 }
{ Current Revision:  30 Oct 1992 }
{          Version:  1.20         }

{ Description:  Contains a procedure used to propagate orbital elements    }
{               (bstar, xincl, e, xnode, omega, xm, xn) and the position   }
{               covariance.  It first propagates for the time prop_time,   }
{               typically the input time of the last observation.  It then }
{               calculates the time of the last ascending node crossing    }
{               and propagates the elements and Phi matrix to that time.   }
{               It then deweights the COV matrix with an exponential factor}
{               and checks if the diag elements of the COV exceed a limit. }
{               If exceeded, the element is reset to the limit.            }
{               This procedure gives only a first order propagation using  }
{               only J2.                                                   }

INTERFACE
Uses DC_Init,                {argument types}
     DC_Out;

Procedure Propagate(var prop_time : double;
                        input_elset : matrix8x1;
                          input_COV : matrix7x7;
                     var prop_elset : matrix8x1;
            var deweighted_prop_COV : matrix7x7);

IMPLEMENTATION
Uses SGP_Intf,               {constants}
     SGP_Math;               {functions Cube & Power}

Procedure Propagate(var prop_time : double;
                        input_elset : matrix8x1;
                          input_COV : matrix7x7;
                     var prop_elset : matrix8x1;
            var deweighted_prop_COV : matrix7x7);

  const
    ae_sqr = ae*ae;                {(Radius of the earth in Distance Unit) squared}
  var
    i,j                  : byte;   {counters}
    epsilon,                       {small # used in expansion of xnz in minutes^(4/3)}
    dndnk,                         {Partial mean motion w.r.t. Kozai mean motion}
    dndeps,                        {Partial mean motion w.r.t. epsilon}
    depsdi,                        {Partial epsilon w.r.t. inclination}
    depsde,                        {Partial epsilon w.r.t. eccentricity}
    nz,                            {Kozai mean motion in radians per minute}
    xke,                           {square root of ge (earth grav constant)}
    eps2,eps3,eps4,                {epsilon squared, cubed, ^4}
    bstar1,xincl1,xnode1,          {input elements}
    e1,omega1,xm1,xn1,             {input elements}
    bstar2,xincl2,xnode2,          {propagated elements at last observation}
    e2,omega2,xm2,xn2,             {propagated elements at last observation}
    nu_d,                          {true anomoly desired (at node crossing)}
    xm_d,                          {mean anomoly desired (at node crossing)}
    bstar3,xincl3,xnode3,          {propagated elements at ascending node}
    e3,omega3,xm3,xn3,             {propagated elements at ascentind node}
    delta_prop_time,               {time since last node passage}
    prop_time_min,                 {time of propagation in minutes}
    memrate,                       {memory "forgetting" rate}
    Weight_Factor        : double;{deweighting factor for COV matrix}
    COV_limit            : vector;{Limits on size of diag elements of COV}
    Phi,
    Phi_COV,
    prop_COV             : matrix7x7;

  begin
    prop_time_min := prop_time * 24 * 60;

  {assign input_elset array to input element names}

    bstar1      := input_elset[1];
    xincl1      := input_elset[2];
    xnode1      := input_elset[3];
    e1          := input_elset[4];
    omega1      := input_elset[5];
```

```
  xm1          := input_elset[6];
  xn1          := input_elset[7];
{define constants}
  xke     := Sqrt(3600*ge/Cube(xkmper));        {DU^3/min^2}
  epsilon := (3*ae_sqr*J2*(1+3*cos(2*xincl1))) /
             (8*Power(1-sqr(e1),(3/2))*Power(xke,(4/3)));
  eps2    := epsilon * epsilon;
  eps3    := epsilon * eps2;
  eps4    := eps2 * eps2;
  nz      := xn1 - epsilon*Power(xn1,(7/3))
                + (7*eps2*Power(xn1,(11/3))) / 3
                - 7*eps3*Power(xn1,5)
                + (1925*eps4*Power(xn1,(19/3))) / 81;
  dndnk   := 1 - (7.0 * epsilon * Power(xn1,(4/3))) / 3.0
              + (77.0 * eps2 * Power(xn1,(8/3))) / 9.0
              - (35.0 * eps3 * Power(xn1,4))
              + (36575.0 * eps4 * Power(xn1,(16/3))) / 243.0;
  dndeps  := - Power(xn1,7/3)
             + (14.0 * epsilon * Power(xn1,(11/3))) / 3.0
             - (21.0 * eps2 * Power(xn1,5))
             + (7700.0 * eps3 * Power(xn1,(19/3))) / 81.0;
  depsdi  := - (9.0*ae_sqr*J2*sin(2*xincl1)) /
              (4.0*Power(1-sqr(e1),(3/2))*Power(xke,(4/3)));
  depsde  :=  (9.0*e1*ae_sqr*J2*(1+3*cos(2*xincl1))) /
              (8.0*Power(1-sqr(e1),(5/2))*Power(xke,(4/3)));
{New Propagated elements at time of last observation}
  bstar2 :=  bstar1;
  xincl2 :=  xincl1;
  xnode2 :=  Fmod2p(xnode1 - (3*ae_sqr*J2*Power(nz,(7/3))*cos(xincl1))*prop_time_min /
                   (2*sqr(1-sqr(e1))*Power(xke,(4/3))));
  e2     :=  e1;
  omega2 :=  Fmod2p(omega1 + (3*ae_sqr*J2*Power(nz,(7/3))*(3+5*cos(2*xincl1)))*prop_time_min /
                   (8*sqr(1-sqr(e1))*Power(xke,(4/3))));
  xm2    :=  Fmod2p(xm1+xn1*prop_time_min);
  xn2    :=  xn1;
{New Propagated elements at time of ascending node passage}
  nu_d := twopi - omega2;       {desired true anomaly at ascending node}
  IF nu_d < pi
    THEN
      xm_d := ArcCos((e2+Cos(nu_d))/(1+e2*Cos(nu_d)))
            - e2*Sin(ArcCos((e2+Cos(nu_d))/(1+e2*Cos(nu_d))))
    ELSE
      xm_d :=twopi - (ArcCos((e2+Cos(nu_d))/(1+e2*Cos(nu_d)))
            - e2*Sin(ArcCos((e2+Cos(nu_d))/(1+e2*Cos(nu_d)))));
  IF xm2 > xm_d
    THEN
      delta_prop_time := (xm_d - xm2) / xn2
    ELSE
      delta_prop_time := (xm_d - xm2 - twopi) / xn2;
  prop_time_min := prop_time_min + delta_prop_time;
  bstar3 :=  bstar2;
  xincl3 :=  xincl2;
  xnode3 :=  Fmod2p(xnode2 - (3*ae_sqr*J2*Power(nz,(7/3))*cos(xincl2))*delta_prop_time /
                   (2*sqr(1-sqr(e2))*Power(xke,(4/3))));
  e3     :=  e2;
  omega3 :=  Fmod2p(omega2 + (3*ae_sqr*J2*Power(nz,(7/3))*(3+5*cos(2*xincl2)))*delta_prop_time /
                   (8*sqr(1-sqr(e2))*Power(xke,(4/3))));
  xm3    :=  Fmod2p(xm2+xn2*delta_prop_time);
  xn3    :=  xn2;
{assign propagated elements to prop_elset array}
  prop_time      := prop_time_min / (24*60);
  prop_elset[0] := input_elset[0] + prop_time;
  prop_elset[1] := bstar3;
  prop_elset[2] := xincl3;
  prop_elset[3] := xnode3;
  prop_elset[4] := e3;
  prop_elset[5] := omega3;
  prop_elset[6] := xm3;
  prop_elset[7] := xn3;
{Initialize the Phi matrix to an Identity Matrix}
```

```
    for i := 1 to 7 do
      for j := 1 to 7 do
        IF i = j THEN
          Phi[i,j] := 1.0
        ELSE
          Phi[i,j] := 0.0;

  {Calculate the non-zero values of the Phi Matrix}

    Phi[3,2] := (3*J2*prop_time_min*ae_sqr*Power(nz,(7/3))*Sin(xincl1)) /
                (2*sqr(1 - sqr(e1))*Power(xke,(4/3)))
              - (7*J2*prop_time_min*ae_sqr*Cos(xincl1)*Power(nz,(4/3))*dndeps*depsdi) /
                (2*sqr(1 - sqr(e1))*Power(xke,(4/3)));
    Phi[3,4] := (-6*e1*J2*prop_time_min*ae_sqr*Cos(xincl1)*Power(nz,(7/3))) /
                (Power(1 - sqr(e1),3)*Power(xke,(4/3)))
              - (7*J2*prop_time_min*ae_sqr*Cos(xincl1)*Power(nz,(4/3))*dndeps*depsde) /
                (2*sqr(1 - sqr(e1))*Power(xke,(4/3)));
    Phi[3,7] := (-7*J2*prop_time_min*ae_sqr*Cos(xincl1)*Power(nz,(4/3))*dndnk) /
                (2*sqr(1 - sqr(e1))*Power(xke,(4/3)));
    Phi[5,2] := (-15*J2*prop_time_min*ae_sqr*Power(nz,(7/3))*Sin(2*xincl1)) /
                (4*sqr(1 - sqr(e1))*Power(xke,(4/3)))
              + (7*J2*prop_time_min*ae_sqr*(3 + 5*Cos(2*xincl1))*Power(nz,(4/3))*dndeps*depsdi) /
                (8*sqr(1 - sqr(e1))*Power(xke,(4/3)));
    Phi[5,4] := (-3*e1*J2*prop_time_min*ae_sqr*(3 + 5*Cos(2*xincl1))*Power(nz,(7/3))) /
                (-2*Power(1 - sqr(e1),3)*Power(xke,(4/3)))
              + (7*J2*prop_time_min*ae_sqr*(3 + 5*Cos(2*xincl1))*Power(nz,(4/3))*dndeps*depsde) /
                (8*sqr(1 - sqr(e1))*Power(xke,(4/3)));
    Phi[5,7] := (7*J2*prop_time_min*ae_sqr*(3 + 5*Cos(2*xincl1))*Power(nz,(4/3))*dndnk) /
                (8*sqr(1 - sqr(e1))*Power(xke,(4/3)));
    Phi[6,7] := prop_time_min;

  {Calculate the propagated COV matrix}

    for i := 1 to 7 do
      for j := 1 to 7 do
        Phi_COV[i,j] := Phi[i,1] * input_COV[1,j]
                      + Phi[i,2] * input_COV[2,j]
                      + Phi[i,3] * input_COV[3,j]
                      + Phi[i,4] * input_COV[4,j]
                      + Phi[i,5] * input_COV[5,j]
                      + Phi[i,6] * input_COV[6,j]
                      + Phi[i,7] * input_COV[7,j];
    for i := 1 to 7 do
      for j := 1 to 7 do
        prop_COV[i,j] := Phi_COV[i,1] * Phi[j,1]
                       + Phi_COV[i,2] * Phi[j,2]
                       + Phi_COV[i,3] * Phi[j,3]
                       + Phi_COV[i,4] * Phi[j,4]
                       + Phi_COV[i,5] * Phi[j,5]
                       + Phi_COV[i,6] * Phi[j,6]
                       + Phi_COV[i,7] * Phi[j,7];

  {Deweight the propagated COV matrix}

      memrate       := 1.0 - power(0.28,(1/LUPI));
      Weight_Factor := sqr(exp(-memrate*-LUPI));
      for i := 1 to 7 do
        for j := 1 to 7 do
          deweighted_prop_COV[i,j] := (prop_COV[i,j] * Weight_Factor);

  {Check for diag variance terms exceeding limit & reset to limit if exceeded}

      for i := 1 to 7 do
        COV_limit[i] := 1.0E-15;
      for i:= 1 to 7 do
        if deweighted_prop_COV[i,i] < COV_limit[i] then
          deweighted_prop_COV[i,i] := COV_limit[i];

end;  {Procedure Prop_Phi}

END.
```

Appendix E. *SATELLITE COMPOSITION ANALYSIS*

E.1 Cumulative Distribution and Frequency Histograms

This section includes the cummulative distribution and histogram frequency plots of all satellites included in the current satellite population as investigated in this research.

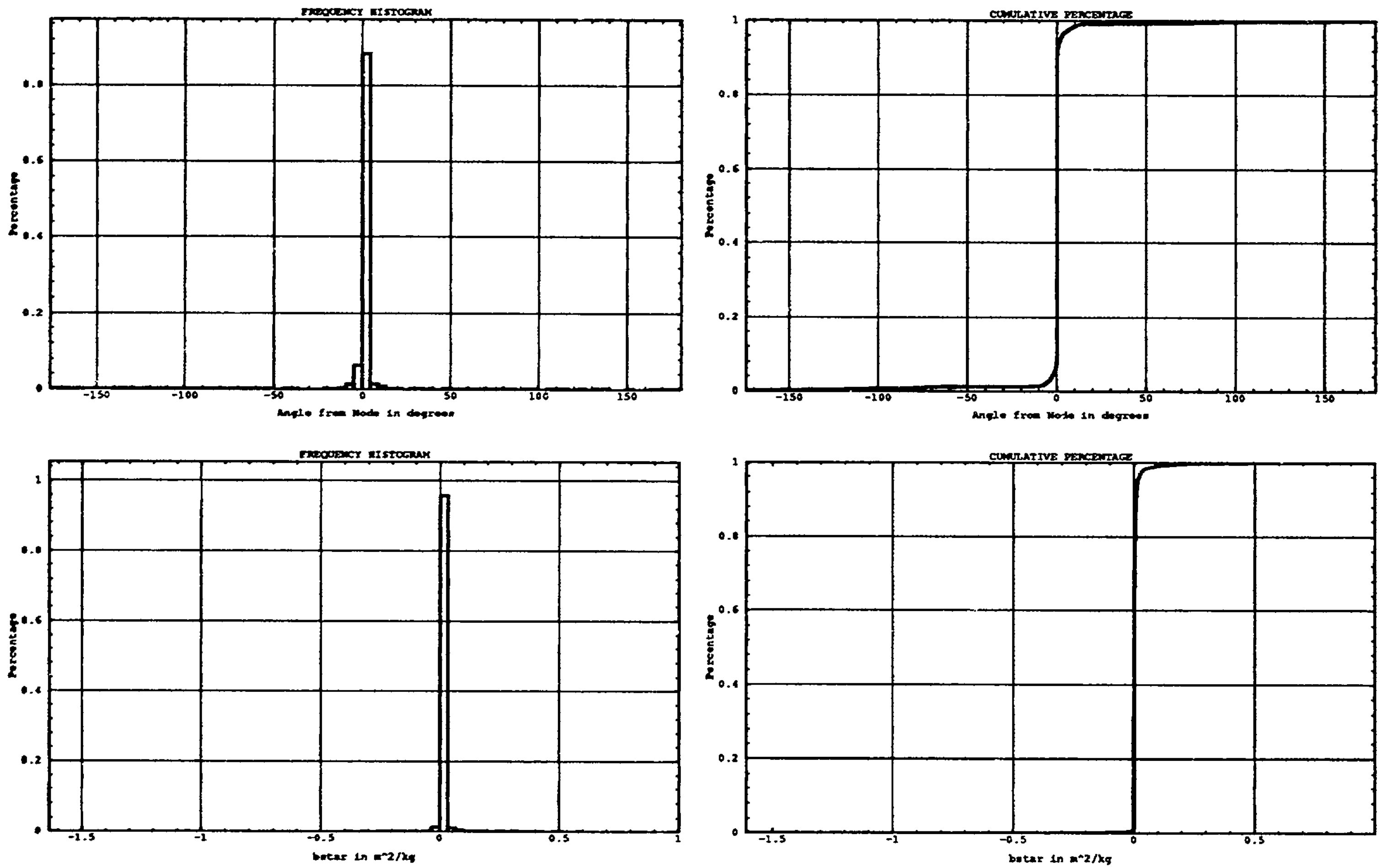


Figure E.1. Histogram and Distribution Plots — Angle From Node, B^*.

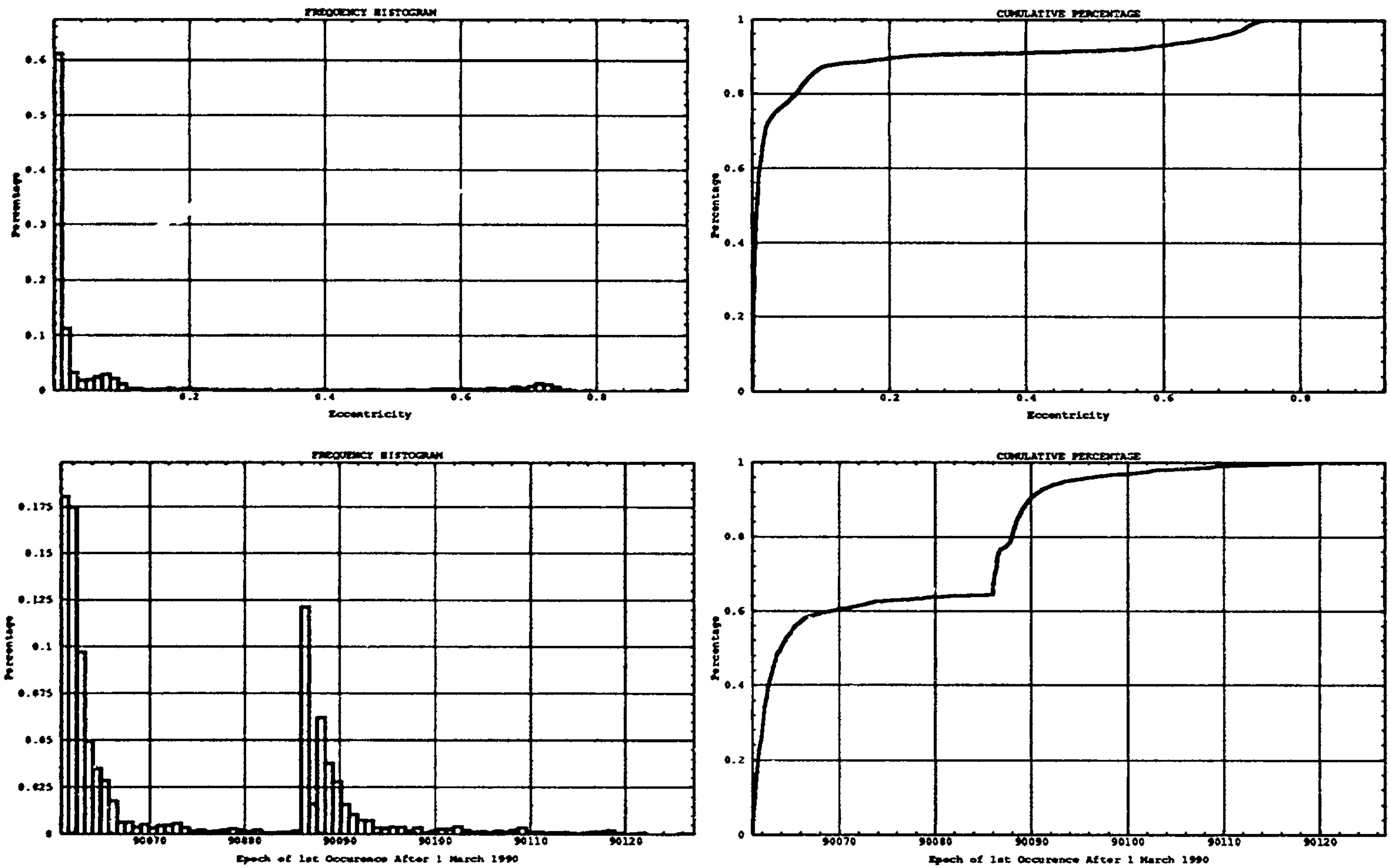


Figure E.2. Histogram and Distribution Plots — Eccentricity, Epoch.

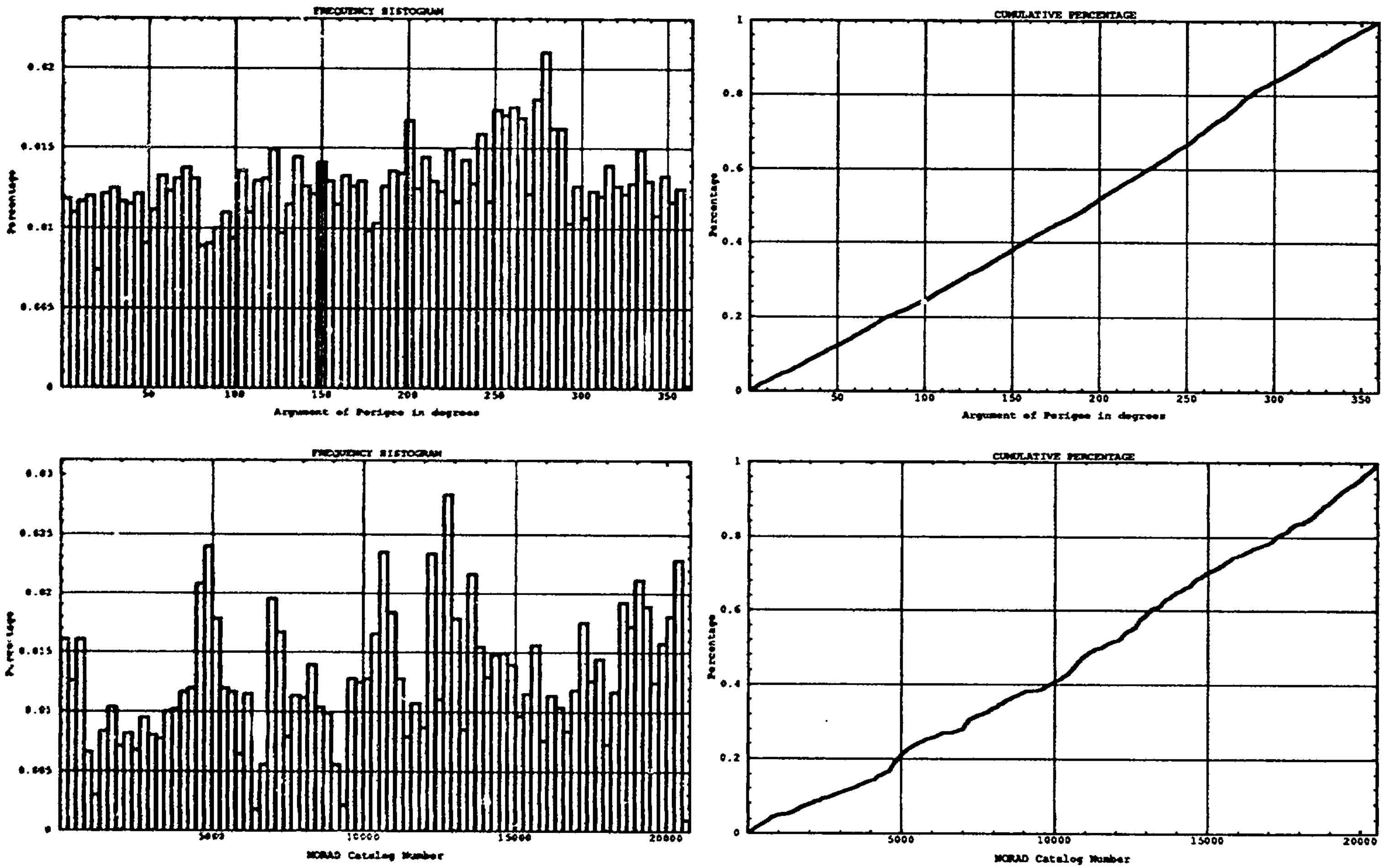


Figure E.3. Histogram and Distribution Plots — Perigee, NORAD Catalog Number.

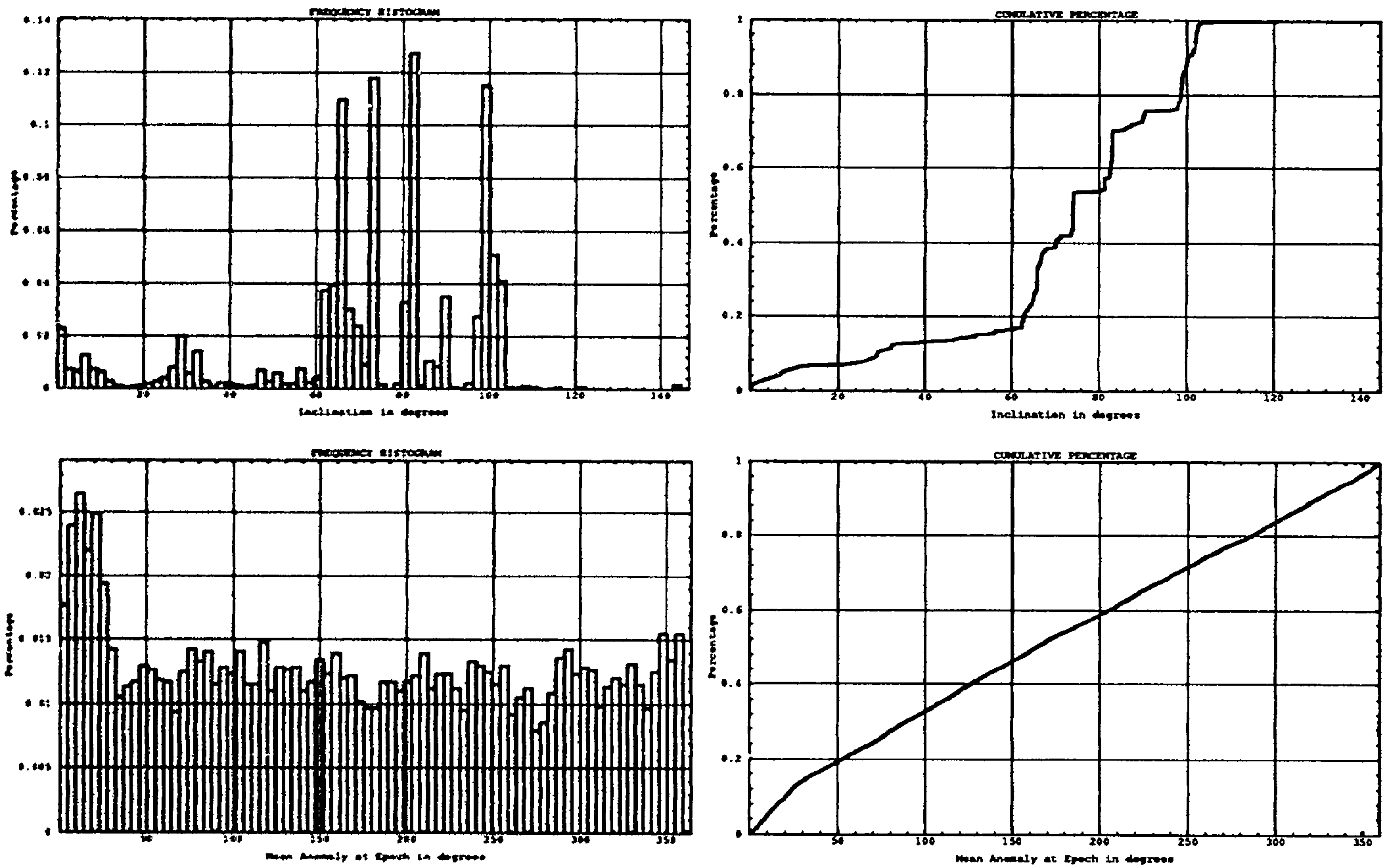


Figure E.4. Histogram and Distribution Plots — Inclination, Mean Anomaly.

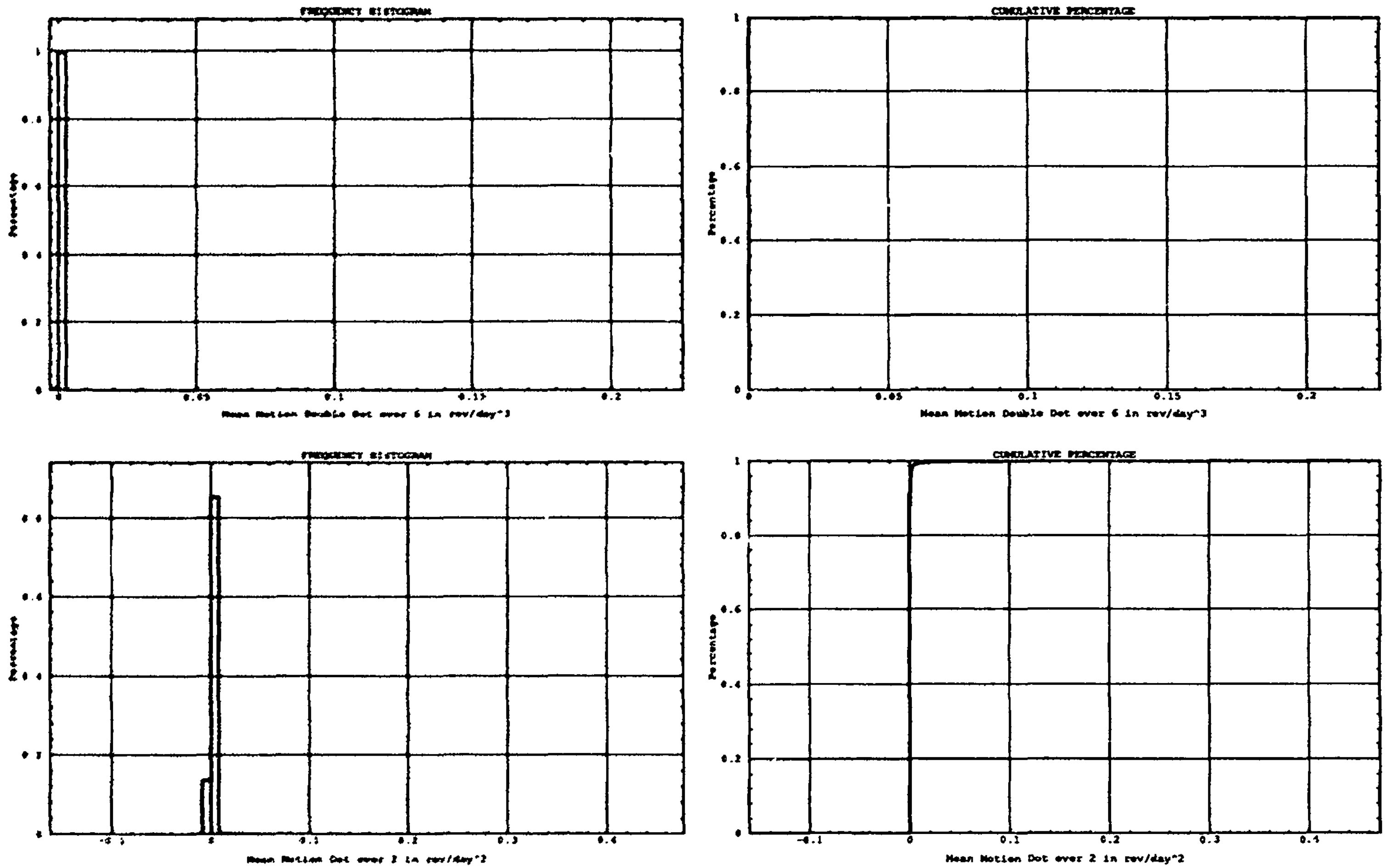


Figure E.5. Histogram and Distribution Plots — Mean Motion Time Derivatives.

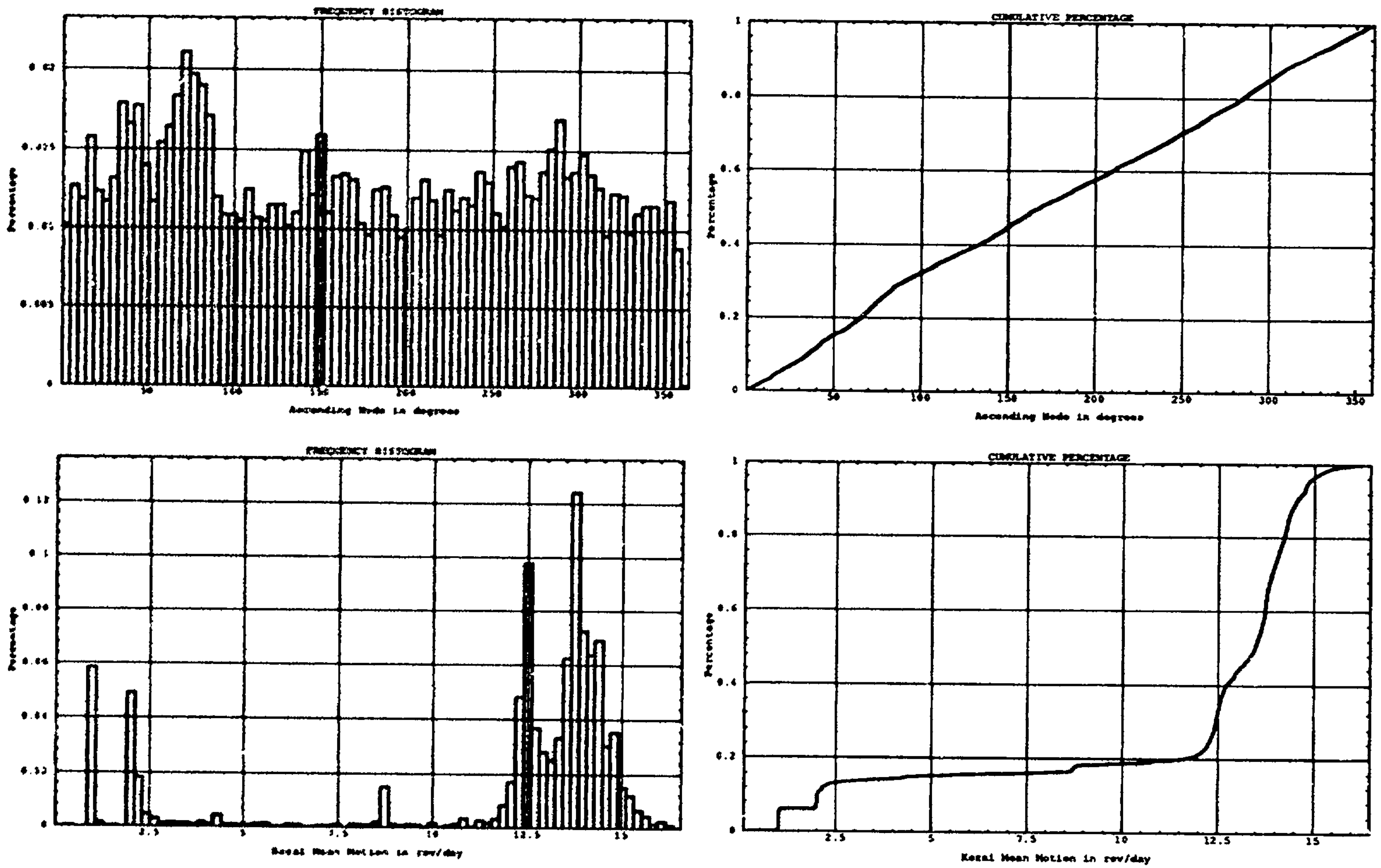


Figure E.6. Histogram and Distribution Plots — Node, Kozai Mean Motion.

E.2 Scatter Plots

Two-variable scatter plots were done for all possible combinations of two-line element set variables. Only plots that were between variables of interest or that showed an unexpected correlation are included in this appendix section or in the body of the text.

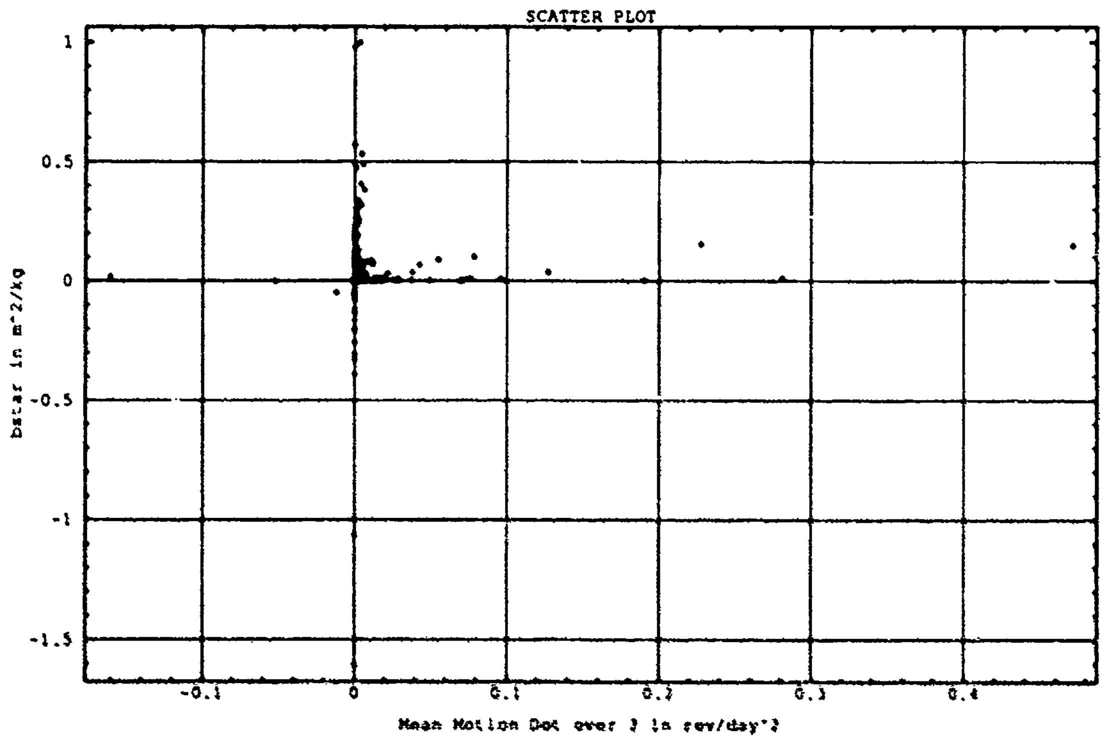


Figure E.7. Scatter Plot — B^* and Mean Motion Dot over 2.

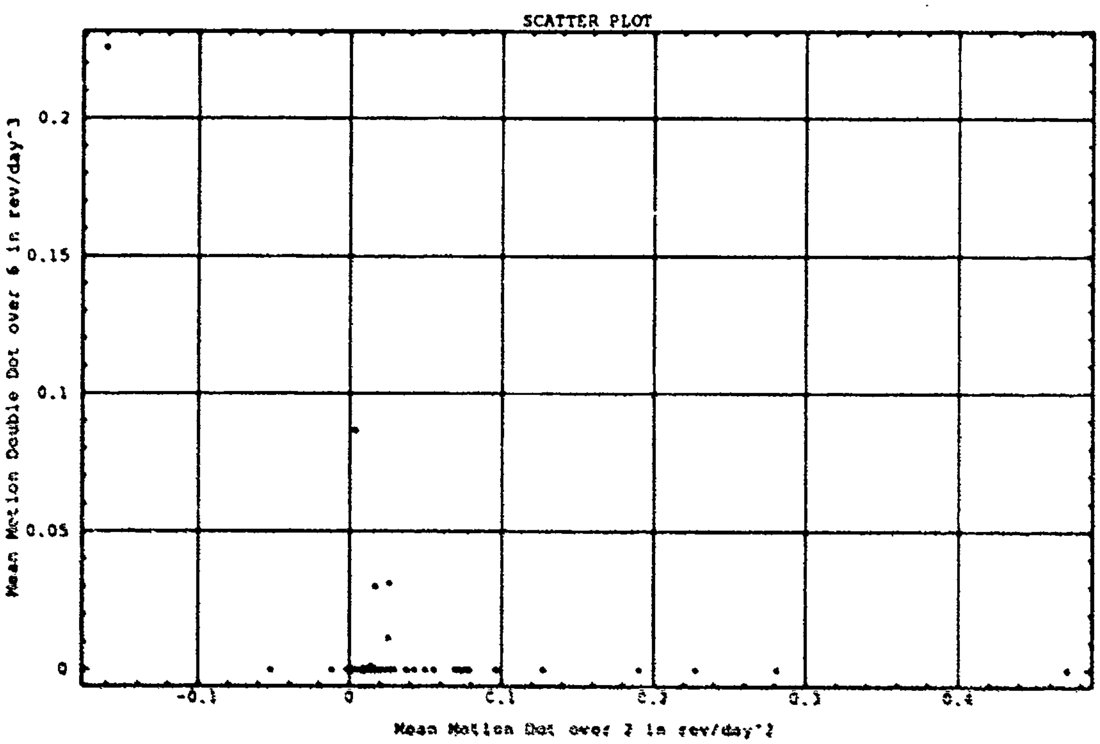


Figure E.8. Scatter Plot — Mean Motion Double Dot over 6 and Mean Motion Dot over 2.

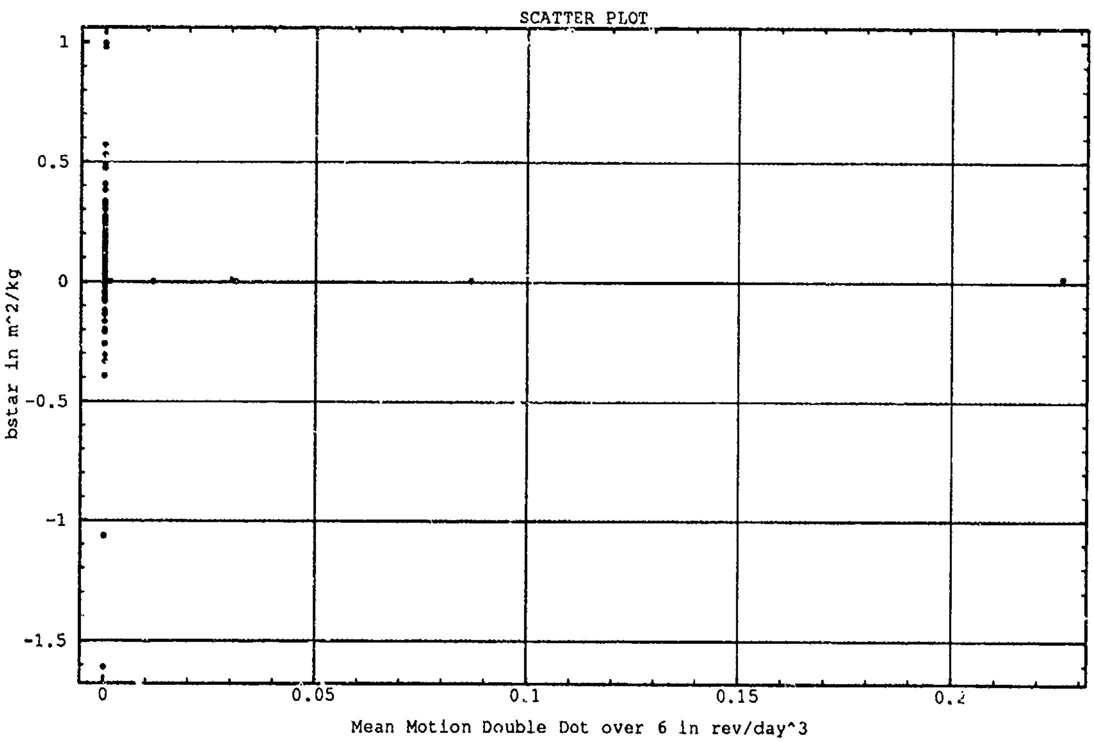


Figure E.9. Scatter Plot — Mean Motion Double Dot over 6 and B^*.

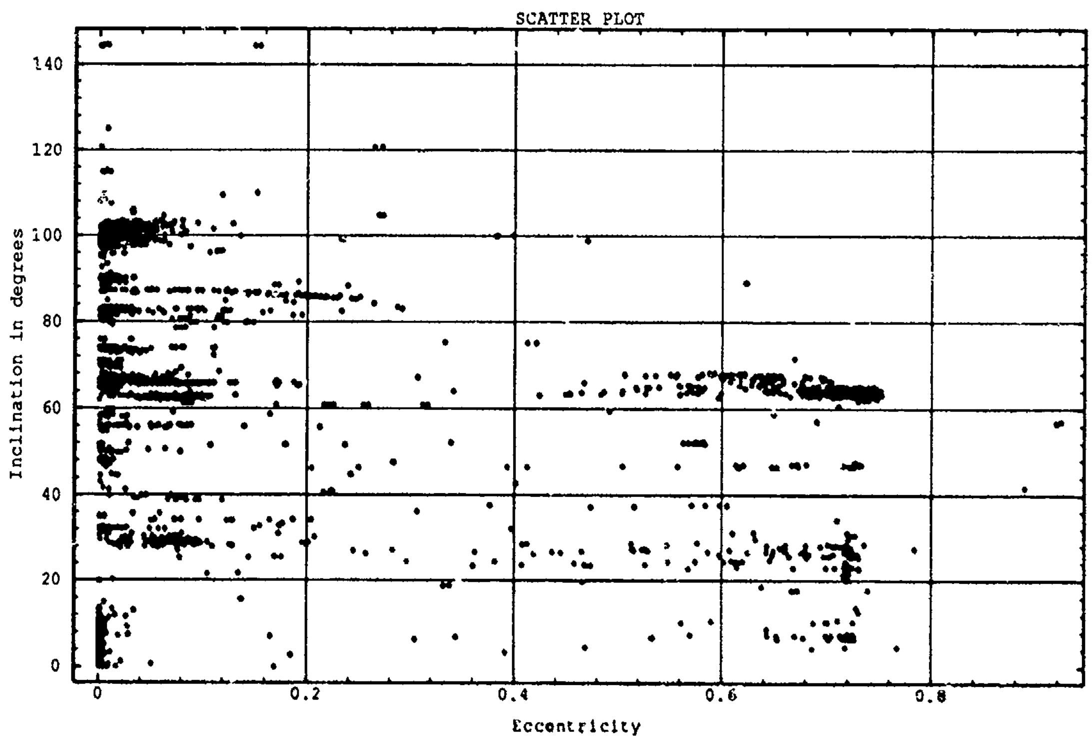


Figure E.10. Scatter Plot — Eccentricity and Inclination.

E.3 Orbital Element Classification Scatter Plots.

Included in this section are the plots showing how the satellites fell into the different Orbital Element Classes. The six plots are for each inclination class. The grid-lines on each plot represent the mean motion and eccentricity combinations.

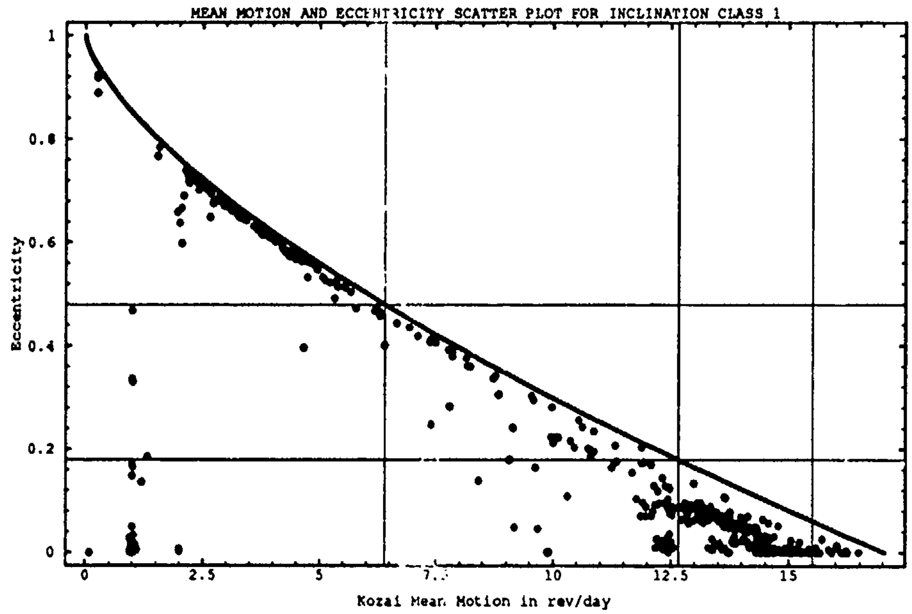


Figure E.11. Classification Breakout — Inclination Class 1.

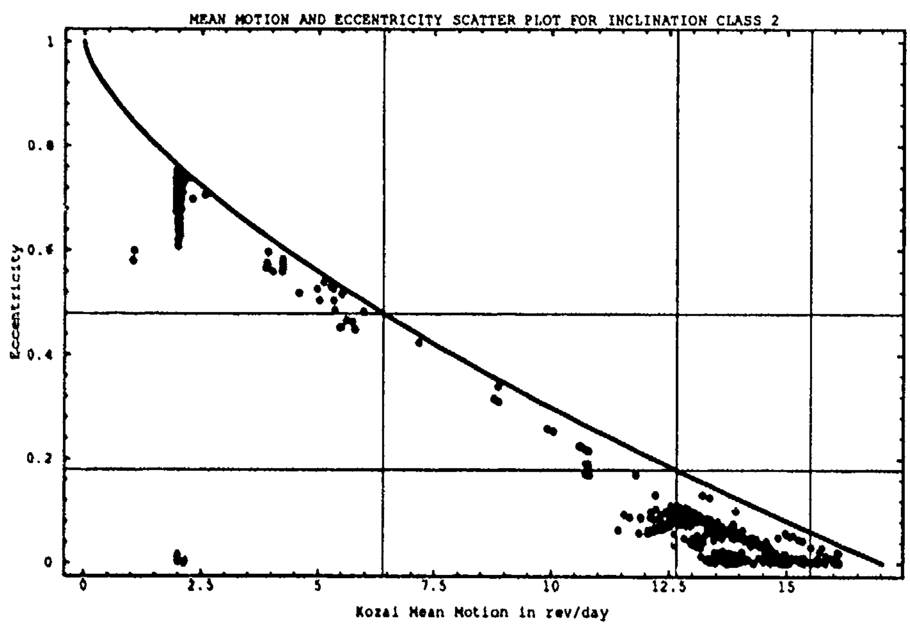


Figure E.12. Classification Breakout — Inclination Class 2.

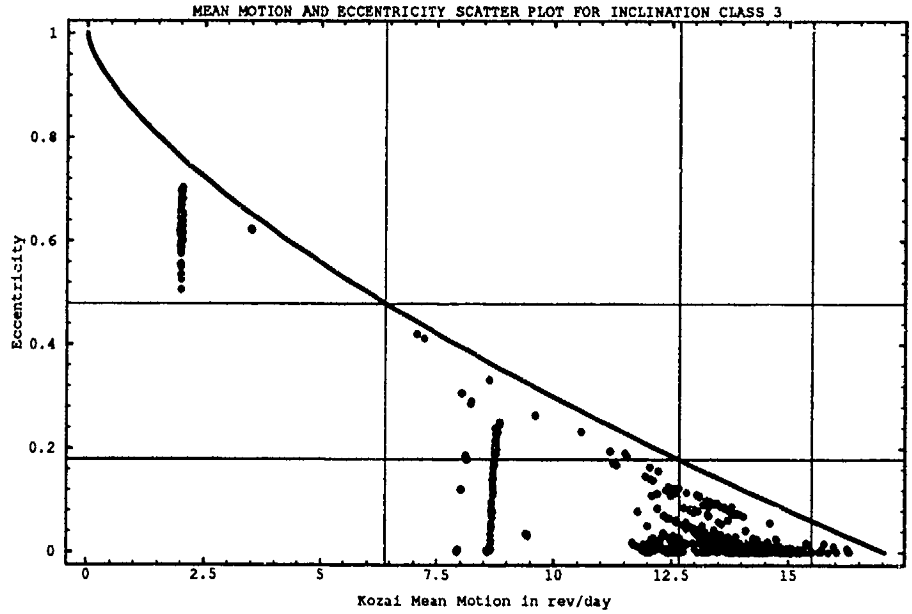


Figure E.13. Classification Breakout — Inclination Class 3.

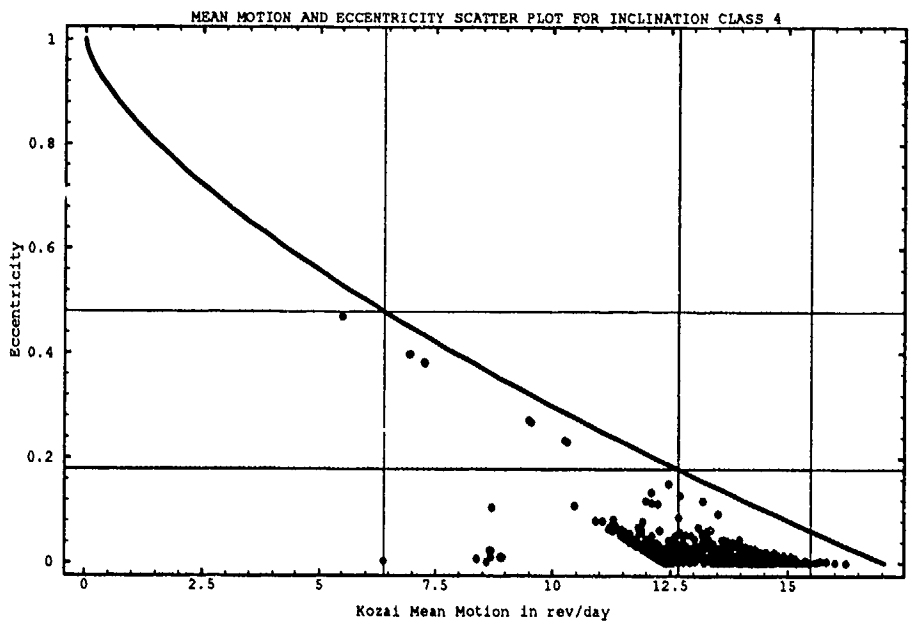


Figure E.14. Classification Breakout — Inclination Class 4.

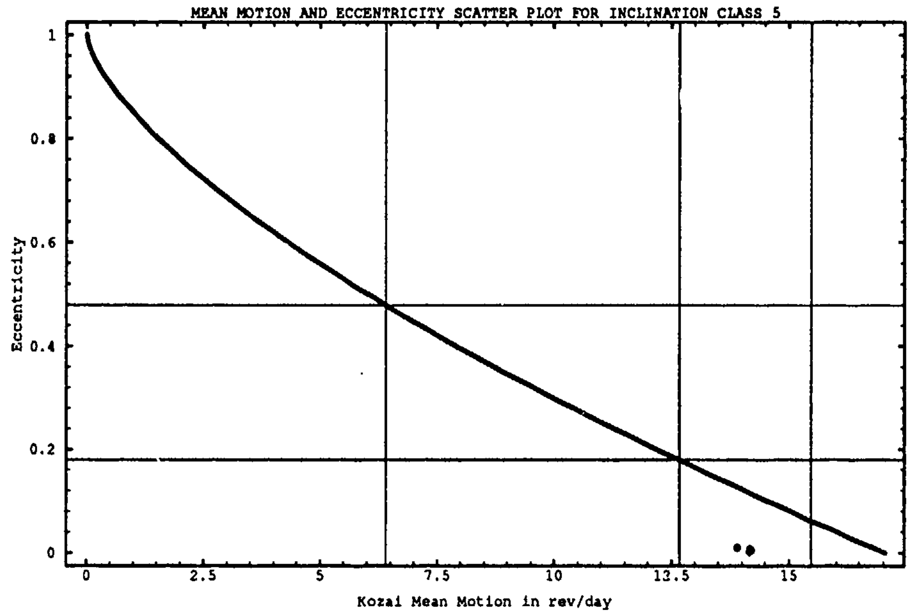


Figure E.15. Classification Breakout — Inclination Class 5.

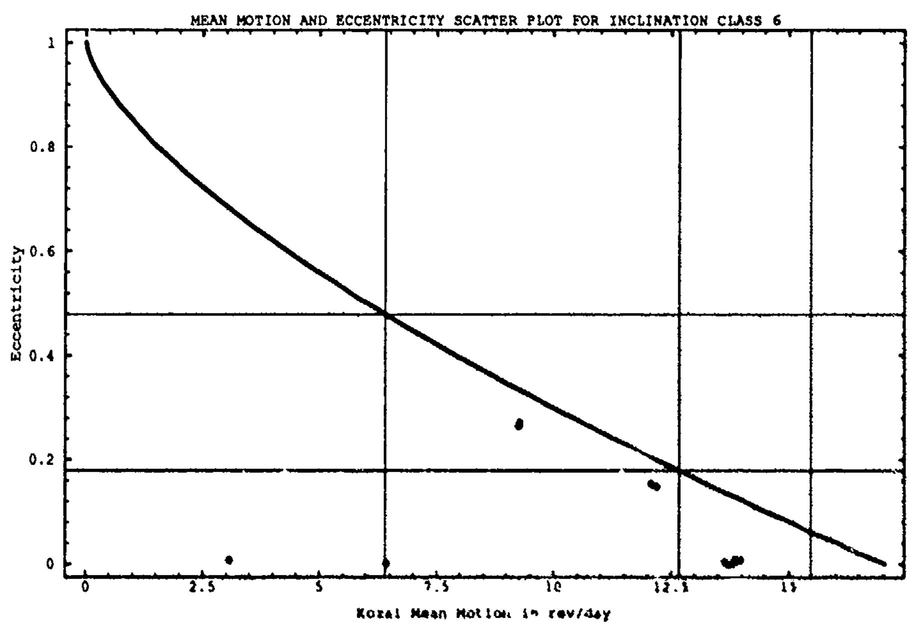


Figure E.16. Classification Breakout — Inclination Class 6.

Appendix F. *SAMPLE SATELLITE SELECTION*

F.1 FORTRAN Source Code used to Divide Satellites into Classes

```
      DOUBLE PRECISION TWOPI, DEGTORAD, EPOCH, XNDT2Z, XINCL
      DOUBLE PRECISION XNODEZ, EZ, OMEGAZ, XMZ, XNZ, XNDD6Z, BSTAR
      DOUBLE PRECISION CLXNZ(1:4), CLEZ(1:3), CLXINCL(1:6)
      INTEGER COUNTER, SATOLD, SATREJ, REJOLD, LINE1, SATNO1
      INTEGER IXNDD6Z, IEXP, IBSTAR, IBEXP, LINE2, SATNO2
      INTEGER I, J, K, UNITNUM
      CHARACTER*30 FILENAME
      CHARACTER*69 TLEPT1, TLEPT2

      DATA CLXNZ/ 0.027925268, 0.0552543, 0.0675843, 0.0743668 /,
     :   CLEZ/ .179665, .47951, 1.00000/,
     :   CLXINCL/ 1.05715, 1.15715, 1.5708, 1.98444, 2.08444, 3.141593/
     :   COUNTER/ 0 /, SATOLD/ 0 /, SATREJ/ 0 /, REJOLD/ 0 /,
     :   TWOPI/ 6.28318530717958647D+0 /

      DEGTORAD = TWOPI/360.0D+0
      OPEN(UNIT=50, FILE='1990.tle', STATUS='OLD')
      OPEN(UNIT=51, FILE='oe9003.fix')
      OPEN(UNIT=52, FILE='firsts_9003.tle')

10    READ (50, 801, END = 999) LINE1, SATNO1, EPOCH, XNDT2Z,
     :           IXNDD6Z, IEXP, IBSTAR, IBEXP
      READ (50, 802, END = 999) LINE2, SATNO2, XINCL, XNODEZ,
     :           EZ, OMEGAZ, XMZ, XNZ

      IF (((LINE1.NE.1).OR.(LINE2.NE.2)).OR.(SATNO1.NE.SATNO2)) THEN
        PRINT *, LINE1, LINE2, SATNO1, SATNO2, EPOCH
        SATREJ = SATNO1
        GOTO 10
      ELSE IF (XNZ .LT. 0.0D+0) THEN
        PRINT *, SATNO1, XNZ
        SATREJ = SATNO1
        GOTO 10
      ELSE IF (DBLE(IBSTAR)*1.0D-5*10.0**(IBEXP) .GT. 1.0D+0) THEN
        PRINT *, SATNO1, IBSTAR*1.0D-5*10.0**(IBEXP)
        SATREJ = SATNO1
        GOTO 10
      ELSE IF (EPOCH .LT. 90061.0) THEN
C         PRINT *, SATNO1, EPOCH
        SATREJ = SATNO1
        GOTO 10
      ELSE IF (SATNO1 .EQ. SATOLD) THEN
        GOTO 10
      ELSE IF (SATNO1 .EQ. SATREJ) THEN
        GOTO 20
      ELSE
        IF (SATREJ .NE. REJOLD) THEN
          PRINT *, SATREJ
          REJOLD = SATREJ
          GOTO 20
        ELSE
          GOTO 20
        ENDIF
      END IF

20    BACKSPACE(UNIT=50)
      BACKSPACE(UNIT=50)
      READ (50, 800, END = 999) TLEPT1
      READ (50, 800, END = 999) TLEPT2

      REJOLD  = SATREJ
      XNDD6Z  = DBLE(IXNDD6Z)*1.0D-5*10.0**(IEXP)
      BSTAR   = DBLE(IBSTAR)*1.0D-5*10.0**(IBEXP)
      XNODEZ  = XNODEZ*DEGTORAD
      OMEGAZ  = OMEGAZ*DEGTORAD
      XMZ     = XMZ*DEGTORAD
      XINCL   = XINCL*DEGTORAD
      XNZ     = XNZ*TWOPI/1440.0D+0
```

```
      XNDT2Z  = XNDT2Z*TWOPI/(1440.0D+0*1440.0D+0)
      XNDD6Z  = XNDD6Z*TWOPI/(1440.0D+0*1440.0D+0*1440.0D+0)
      SATOLD  = SATNO1
      COUNTER = COUNTER + 1

      I = 1
      J = 1
      K = 1
30    IF (XNZ .GT. CLXNZ(I)) THEN
        I = I + 1
      GO TO 30
      END IF
40    IF (EZ .GT. CLEZ(J)) THEN
        J = J + 1
      GO TO 40
      END IF
50    IF (XINCL .GT. CLXINCL(K)) THEN
        K = K + 1
      GO TO 50
      END IF

      FILENAME = 'oe9003_xnz' // CHAR(48+I) // '_ez' //
     :             CHAR(48+J) // '_xincl' // CHAR(48+K) // '.fix'
      UNITNUM = 100*I + 10*J + 1*K
      OPEN(UNIT= UNITNUM, FILE = FILENAME)
      WRITE(UNITNUM, 803) EPOCH, COUNTER, SATNO1,
     :                    XNZ,   XNDT2Z,  XNDD6Z,
     :                    XINCL, XNODEZ,  OMEGAZ,
     :                    EZ,    BSTAR,   XMZ
      WRITE(51, 803) EPOCH, COUNTER, SATNO1,
     :                    XNZ,   XNDT2Z,  XNDD6Z,
     :                    XINCL, XNODEZ,  OMEGAZ,
     :                    EZ,    BSTAR,   XMZ
      WRITE(52, 800) TLEPT1
      WRITE(52, 800) TLEPT2

      GOTO 10

800   FORMAT (A69)
801   FORMAT (I1, 1X, I5, 11X, F14.8, 1X, F10.8, 1X, I6, I2, 1X, I6, I2)
802   FORMAT (I1, 1X, I5, 1X, 2(F8.4, 1X), F7.7,1X, 2(F8.4, 1X), F11.8)
803   FORMAT(F14.8,1X,I5,1X,I5,3(/,3(1X,E15.8)))

999   CLOSE(50)
      CLOSE(51)
      CLOSE(52)

      DO 901, I = 1,4,1
      DO 901, J = 1,3,1
      DO 901, K = 1,6,1
      UNITNUM = 100*I + 10*J + 1*K
      CLOSE(UNITNUM)
901   CONTINUE

      END
```

F.2 Example of FORTRAN Output

file oe9003_xnz1_ez1_xincl6

```
90061.08689731  1901  7369
  0.13404702E-01  0.60601710E-12  0.00000000E+00
  0.21824330E+01  0.48838449E+01  0.65970479E+00
  0.81123000E-02  0.00000000E+00  0.56344499E+01
            .
            .
            .
            .
```

F.3 Satellites that Failed Satellite Division Criteria

Satellites that did not have a two-line elenent set in our data file after 1 March 1990 but did have an entry between 1 January and 28 February 1990.[1]

02473

Satellites that had $B^* > 1.0$ on first and possibly subsequent occurrances of two-line element set entries after 28 February, but did have at least one entry with a $B^* < 1.0$.[2]

00579	02372	02933	05316	06192	08785	09330	09468	09635
11550	16103	18615	18672	18995	19164	19394	19452	19502
19644	19751	19964	20004	20044	20201	20454		

Satellites that had $B^* > 1.0$ on first and possibly subsequent occurrances of two-line element set entries after 28 February, and did not have one entry with a $B^* < 1.0$.[3]

11888 12078 14182 15147 18231

F.4 Mathamatica Function for Sample Satellite Selection

```
<<Statistics'DescriptiveStatistics'; <<Statistics'ContinuousDistributions';
<<Statistics'DataManipulation';

(*     This function selects one 2-line OE set with the least          *)
(*     deviation from the mean, using the elements specified in Vmean  *)
(*     below.                                                          *)
(*     Example useage:                                                 *)
(*             select["~/math/oe/oe9003_xnz1_ez1_xincl1.fix"]          *)
(*     Limitations:                                                    *)
(*             It requires more than one 2-line OE set to choose from. *)
(*             It chooses the second 2-line OE set if there are only   *)
(*                     two to choose from.                             *)

select[filename_] := Block[ {oe, epoch, counter, satno, xnz, xndt2z, xndd6z,
  xincl, xnodez, omegaz, ez, bstar, xmz, Vmean, Vstdev, partoe, expon, sel},
    oe      = ReadList[filename, Table[Number, {12}]];
    epoch   = Transpose[oe][[1]];
    counter = Transpose[oe][[2]];
    satno   = Transpose[oe][[3]];
    xnz     = Transpose[oe][[4]];
    xndt2z  = Transpose[oe][[5]];
    xndd6z  = Transpose[oe][[6]];
    xincl   = Transpose[oe][[7]];
    xnodez  = Transpose[oe][[8]];
    omegaz  = Transpose[oe][[9]];
    ez      = Transpose[oe][[10]];
    bstar   = Transpose[oe][[11]];
    xmz     = Transpose[oe][[12]];
    Vmean   = {Mean[xnz], Mean[xndt2z], Mean[xincl], Mean[xndd6z],
               Mean[xnodez], Mean[omegaz], Mean[ez], Mean[bstar]};
    Vstdev  = {StandardDeviation[xnz], StandardDeviation[xndt2z],
```

[1]These satellites were excluded from further analysis.

[2]These satellites were included in further analysis for the TLE entry with $B^* < 1.0$.

[3]These satellites were excluded from further analysis.

```
            StandardDeviation[xincl], StandardDeviation[xndd6z],
            StandardDeviation[xnodez], StandardDeviation[omegaz],
            StandardDeviation[ez], StandardDeviation[bstar]};
  partoe  = {xnz, xndt2z, xincl, xndd6z, xnodez, omegaz, ez, bstar};
  expon   = Product[If[Vstdev[[i]] > 0, PDF[ NormalDistribution[
            Vmean[[i]], Vstdev[[i]]], partoe[[i]]] * (Sqrt[2Pi] *
            Vstdev[[i]]), 1], {i,8}];
  sel     = Last[Sort[Transpose[ {expon, epoch, counter, satno, xnz, xndt2z,
            xndd6z, xincl, xnodez, omegaz, ez, bstar, xmz}] ] ];
 {sel[[2]],sel[[3]],sel[[4]],sel[[5]],sel[[6]],sel[[7]],sel[[8]],sel[[9]],
 sel[[10]],sel[[11]],sel[[12]]}
 ]
```

F.5 Example of Mathematica Input for Sample Satellite Selection

```
In[2]:= select["~/math/oe/oe9003_xnz1_ez1_xincl1.fix"]

                                                      -12
Out[2]= {90089.8, 4314, 15141, 0.00425943, -2.06046 10    , 0., 0.08716,

>    1.26758, 3.41056, 0.0012258, 0.000099999, 2.87378}
```

F.6 Selected Sample Satellite Two-Line Element Sets

```
CLASS: 1-1-1
  1 15141U 83 66   F 90089.84366136 -.00000068  00000-0  99999-4 0 06562
  2 15141   4.9939  72.6272 0012258 195.4106 164.6556  0.97618901  8101

CLASS: 1-1-2
  1 15259U 84 95   A 90104.43454589  .00000021  00000-0  14999-1 0 06436
  2 15259  64.7864 162.4815 0018219 201.8083 158.2010  2.13103289 43653

CLASS: 1-1-4
  1 08822U 76039   C 90062.13597810  .00000003  00000-0  99999-4 0 05949
  2 08822 109.8562 320.0595 0042402 261.2019  98.3915  6.38859342322640

CLASS: 1-1-6
  1 07369U 74 54   A 90061.08689731  .00000020  00000-0  00000+0 0 09770
  2 07369 125.0442 279.8237 0081123  37.7983 322.8302  3.07213139175423

CLASS: 1-2-1
  1 12497U 81 50   B 90088.32850727  .00016164  00000-0  36133-2 0 03019
  2 12497  23.9029 101.2132 4582441 111.9356 302.0469  6.29011949171514

CLASS: 1-2-2
  1 12827U 81 88   F 90089.41666294 -.00000000  00000-0  99999-4 0 09685
  2 12827  63.3458 225.1199 4536648 331.3832   9.8124  5.48793671171338

CLASS: 1-2-4
  1 16614U 86 19   B 90121.16504499  .00000058  00000-0  51294-3 0 01217
  2 16614  98.8247  43.7071 4697651 253.3377  51.5618  5.50352988 84146

CLASS: 1-3-1
  1 13958U 67   1 AL 90089.18539481  .00009294  00000-0  60407-2 0 08470
  2 13958  26.8024 177.9088 6611432 174.3619 200.5635  3.11659651113951
```

```
CLASS: 1-3-2
 1 14199U 83 73   A 90089.01439699  .00000107  00000-0  21689-2 0 04476
 2 14199  64.0314 151.8929 6984908 267.2024  18.2015  2.30522784 52889

CLASS: 1-3-3
 1 14041U 83 38   E 90094.88172405 -.00000073  00000-0  99999-4 0 03588
 2 14041  66.7266 183.3419 6291201 269.0441  23.4762  2.03595797 51648

CLASS: 2-1-1
 1 00053U 60IOT  5 90062.60444147 -.00000024  00000-0  99999-4 0 02939
 2 00053  47.2785 159.8122 0099598 163.6009 196.8084 12.16554951321254

CLASS: 2-1-2
 1 10440U 76067   R 90061.69524654  .00000769  00000-0  24872-2 0 04444
 2 10440  65.6201 178.4067 0731810 171.0166 190.4830 12.37045978615841

CLASS: 2-1-3
 1 10293U 77 79   J 90066.05843306  .00000005  00000-0  99999-4 0 07414
 2 10293  74.0232 211.3035 0135375 191.4952 168.2984 12.25140528337691

CLASS: 2-1-4
 1 10393U 74089 EJ 90062.62858901 -.00000005  00000-0  99999-4 0 06127
 2 10393 101.5836 125.5560 0159384 186.4464 173.4562 12.23222517688950

CLASS: 2-1-6
 1 03307U 68 55   A 90061.30107875 -.00000004  00000-0  99999-4 0 03273
 2 03307 120.8360  49.7361 0016628 304.3898  55.5064  6.42179368507994

CLASS: 2-2-1
 1 14670U 84  8   A 90089.71771437  .00003989  00000-0  22435-2 0 04016
 2 14670  36.1314 231.2740 3062374 228.4523 101.0035  8.83565400199084

CLASS: 2-2-2
 1 19990U 64  6   T 90061.50986469  .00036262  00000-0  64575-2 0 03393
 2 19990  60.7998 107.0493 2240370 111.3711 273.9890 10.63979039 33063

CLASS: 2-2-3
 1 19859U 63 14 EQ 90061.33277128  .00001585  00000-0  53835-1 0 01474
 2 19859  85.7520 193.3073 2180139 351.0306   5.6905  8.77440823 12684

CLASS: 2-2-4
 1 03825U 69 25   C 90064.05404288  .00001522  00000-0  75393-3 0 04030
 2 03825 104.7482 291.9495 2730748 143.1954 2[illegible].1864  9.47917380721352

CLASS: 2-2-6
 1 04841U 68 55   D 90074.10938230  .00001193  00000-0  15271-2 0 02071
 2 04841 120.7474 345.7001 2717040 280.3514  58.7999  9.26652751199759

CLASS: 3-1-1
 1 01996U 65096   D 90062.57233723  .00029976  00000-0  57772-2 0 01333
 2 01996  34.2488 168.4994 0537978 239.8495 114.7699 13.98433357202306

CLASS: 3-1-2
 1 14443U 77121 BF 90086.44518818  .00010006  00000-0  49856-2 0 09526
 2 14443  65.6376 243.2938 0224462 174.3899 186.0178 14.05013696441828

CLASS: 3-1-3
 1 19643U 78121   C 90062.46477114  .00000628  00000-0  32995-3 0 01914
 2 19643  81.2328 216.8294 0067007 164.6783 195.6607 14.11971226 70214
```

```
CLASS: 3-1-4
  1 17429U 86 19 FQ 90086.10115666  .00012418  00000-0  68353-2 0 04994
  2 17429  98.9992 173.5282 0109512 173.9157 186.3401 14.08678078172316

CLASS: 3-1-5
  1 07735U 75 27  B 90061.92256864  .00000817  00000-0  44554-3 0 05202
  2 07735 114.9854  32.3886 0049798 313.3500  46.3487 14.20662747771802

CLASS: 3-1-6
  1 02327U 66 63  B 90063.84215319  .00003471  00000-0  30012-2 0 07997
  2 02327 144.2199  82.3338 0017522 175.4422 184.6370 13.82228492187809

CLASS: 4-1-1
  1 14002U 83 33  A 90086.44246605  .00466655  83983-4  16472-2 0 07221
  2 14002  46.5599 241.9884 0070169 291.6506  67.7970 15.84324476380871

CLASS: 4-1-2
  1 13972U 83 27  A 90088.25195735  .00192985  00000-0  92495-3 0 04456
  2 13972  65.8266 153.4866 0009504 229.4544 130.5459 15.81905388390904

CLASS: 4-1-3
  1 16438U 85121  E 90086.00040288  .00580501  00000-0  39896-2 0 03141
  2 16438  70.9442 188.7777 0064322 113.8059 246.9918 15.70202629231132

CLASS: 4-1-4
  1 16036U 79 17  P 90092.07768260  .00236728  00000-0  22353-2 0 03377
  2 16036  98.1206 141.8874 0023457 193.5278 163.4477 15.65936551614361

CLASS: 4-1-2 (2nd Choice)
  1 15584U 85 18  A 90086.75596273  .00071118  00000-0  95413-3 0 02527
  2 15584  65.8387 217.5705 0023397 269.2089  90.6387 15.57222498283684

CLASS: 4-1-3 (2nd Choice)
  1 04497U 70 64  A 90062.21980962  .00093972  00000-0  41670-3 0 08423
  2 04497  74.0185 133.5216 0004296 347.6032  12.4936 15.84015404 91061

CLASS: 4-1-1 (2nd Choice)
  1 20335U 89 93  A 90061.64390907  .00009759  00000-0  12823-3 0 01290
  2 20335  51.6166 147.1234 0017535 227.5475 132.5147 15.59068899 15116
```

Appendix G. *MATHEMATICA CODE FOR PLOTS*

The code below was used to plot the VMAGT points for each of the LUPI/OPD iterations using all 10 random observation files for a specific satellite.

```
<<Statistics'DescriptiveStatistics'; <<Statistics'DataManipulation';
<<Statistics'MovingAverage'; <<Statistics'ConfidenceIntervals';
(*      This function creates a graph of a residuals file (vmatrix).   *)
(*      It computes a moving average over four days.  Steady-state is  *)
(*      manually declared at "maxday", and "scale" is the duration of  *)
(*      data span. It also outputs the graph info to file "CI_FILE".   *)
(*      Example usage:                                                 *)
(*            eplot[14443,f88m,4,28,60]                                *)
eplot[satno_Integer,vmatrix_,LUPI_Integer,maxday_Integer,scale_Integer] :=
  Block[ {nruns, vmt, opd, lave, move, vmagt, avp, ave, std, p99, aveCI,
  varCI, p99CI, aver1, aver2, varr1, varr2, p99r1, p99r2, top},
    nruns   = Length[vmatrix];
    vmt     = Sort[ColumnJoin[vmatrix[[1]], vmatrix[[2]], vmatrix[[3]],
                vmatrix[[4]], vmatrix[[5]], vmatrix[[6]], vmatrix[[7]],
                vmatrix[[8]], vmatrix[[9]], vmatrix[[10]]]];
    opd     = Length[vmt]/(nruns*scale);
    lave    = Table[Sum[vmt[[i]], {i,(j-1)*opd+1,j*opd}]/(opd),
                {j,1,Length[vmt]/(opd)}];
    move    = ListPlot[MovingAverage[lave,nruns*4-1],
                PlotJoined -> True, DisplayFunction -> Identity];
    vmagt   = ListPlot[vmt, Prolog->PointSize[.005],
                PlotJoined -> False, DisplayFunction -> Identity];
    avp     = Table[trimstuff[vmatrix[[i]],maxday],{i,1,10}];
    ave     =       Mean[Transpose[avp][[1]] ];
    std     = Sqrt[ Mean[Transpose[avp][[2]] ] ];
    p99     =       Mean[Transpose[avp][[3]] ];
    aveCI   = MeanCI[Transpose[avp][[1]] ];
    varCI   = MeanCI[Transpose[avp][[2]] ];
    p99CI   = MeanCI[Transpose[avp][[3]] ];
    aver1   = Round[100*aveCI[[1]]]/100.0;
    aver2   = Round[100*aveCI[[2]]]/100.0;
    varr1   = Round[100*varCI[[1]]]/100.0;
    varr2   = Round[100*varCI[[2]]]/100.0;
    p99r1   = Round[100*p99CI[[1]]]/100.0;
    p99r2   = Round[100*p99CI[[2]]]/100.0;
    top     = Floor[5/3*p99] + 1;
  OpenAppend["CI_FILE"]; PutAppend[{satno,LUPI,opd,aver1,aver2,varr1,
    varr2,p99r1,p99r2},"CI_FILE"]; Close["CI_FILE"];
  Show[move, vmagt, Prolog -> {GrayLevel[.5], Thickness[.0035],
    Line[{{maxday,0}, {maxday,1000}}], GrayLevel[0],
    Line[{{maxday,p99}, {scale,p99}}], Thickness[.005], PointSize[.005],
    Text["steady-state declared",{maxday-scale/45, .70top}, {-1,0}, {0,1}],
    Text["MLE 99% CL",{(maxday+scale)/2, p99+top/30},{0,1}]},
    Epilog -> { GrayLevel[0],
    Text["95% CI 99% CL "TextForm[p99r1], {.9scale,top*58.5/60},{1,0}],
    Text[" -"TextForm[p99r2],{.9scale,top*58.5/60},{-1,0}],
    Text["95% CI mean "  TextForm[aver1], {.9scale,top*56.5/60},{1,0}],
    Text[" -"TextForm[aver2],{.9scale,top*56.5/60},{-1,0}],
    Text["95% CI var. "  TextForm[varr1], {.9scale,top*54.5/60},{1,0}],
    Text[" -"TextForm[varr2],{.9scale,top*54.5/60},{-1,0}]          },
    Frame->True, Axes -> False, PlotRange -> {{-0,scale},{-0,top}},
    GridLines-> {Table[LUPI*i,{i,0,Floor[scale/LUPI] + 1}], Automatic},
    FrameLabel -> {"Time since initial epoch in days", "Position error in km"},
    PlotLabel -> opd"OBS/DAY - TRUE POSITION ERROR AND 4 DAY MOVING AVERAGE",
    DisplayFunction -> $DisplayFunction]
  ]
(*      Computes average CI and MLE variance after removing mdays.       *)
trimstuff[vmt_,mdays_] := Block[{trim, ave, var, p99},
    trim  = BooleanSelect[vmt, Table[vmt[[i,1]]>=mdays, {i,Length[vmt]}] ],
    ave   = Mean[Column[trim,2]];
    var   = VarianceMLE[Column[trim,2]];
    p99   = ave + 2.32635*Sqrt[var];
  {ave, var, p99}
  ]
data[filename_]    := ReadList[filename, Table[Number, {2}]],
```

Appendix H. *VMAGT GRAPHS*

H.1 CLASS: 1-1-1 (NORAD Catalog Number 15141)

H.1.1 Prediction Performance. The tables below provide the statistical information on the ability of the estimates to predict for 1 LUPI after differential correction.

Table H.1. 95% Confidence Interval Analysis on Class: 1–1–1.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
15141	2	2	21.64	31.46	487.95	1404.68	72.85	116.61
15141	2	4	21.99	28.80	530.10	1155.96	75.26	106.76
15141	2	6	21.00	28.41	455.47	978.87	70.75	100.01
15141	2	8	22.03	25.82	467.74	864.38	72.84	93.13
15141	4	2	29.39	47.00	86.08	2031.48	69.17	141.68
15141	4	4	27.75	33.51	339.87	775.17	72.27	95.77
15141	4	6	24.66	29.58	232.68	499.46	60.29	80.48
15141	4	8	23.47	28.31	239.09	443.33	60.59	75.74
15141	6	2	38.38	54.89	474.79	1067.75	89.09	129.23
15141	6	4	35.31	48.29	422.10	1328.83	83.38	130.76
15141	6	6	34.69	41.05	308.38	663.55	75.21	100.16
15141	6	8	34.46	38.45	299.10	525.48	74.64	91.29
15141	8	2	44.77	65.60	358.74	994.31	90.71	135.61
15141	8	4	39.58	52.65	274.12	482.20	79.47	101.96
15141	8	6	37.40	47.70	263.32	386.98	79.02	89.39
15141	8	8	38.27	45.63	245.57	324.44	75.84	86.28

Table H.2. ANOVA Analysis on Class: 1-1-1.

Source of Variation	Degrees of Freedom	Sum of Square	Mean Square	F	Table F
Replications	9	10023.69	1113.74	2.44	1.88
Main Effects:					
LUPI	3	4653.63	1551.21	3.39	2.60
OPD	3	17804.13	5934.71	12.99	2.60
Interaction	9	3459.18	384.53	0.8418	1.88
Error	135	61638.12	456.58		
Total	159	97578.75			

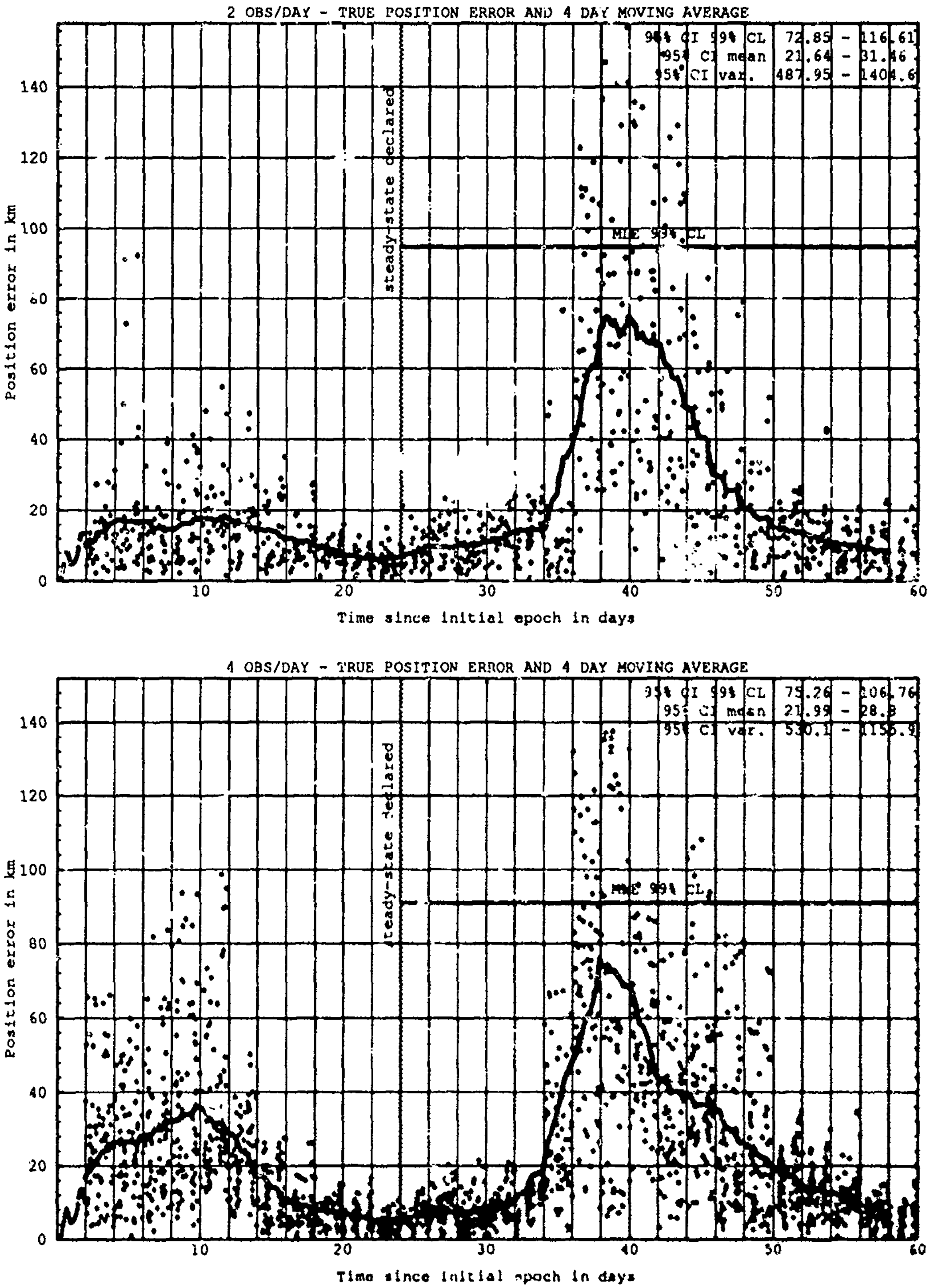


Figure H.1. Class: 1-1-1 (Catalog Number 15141), LUPI 2, OPD 2 and 4.

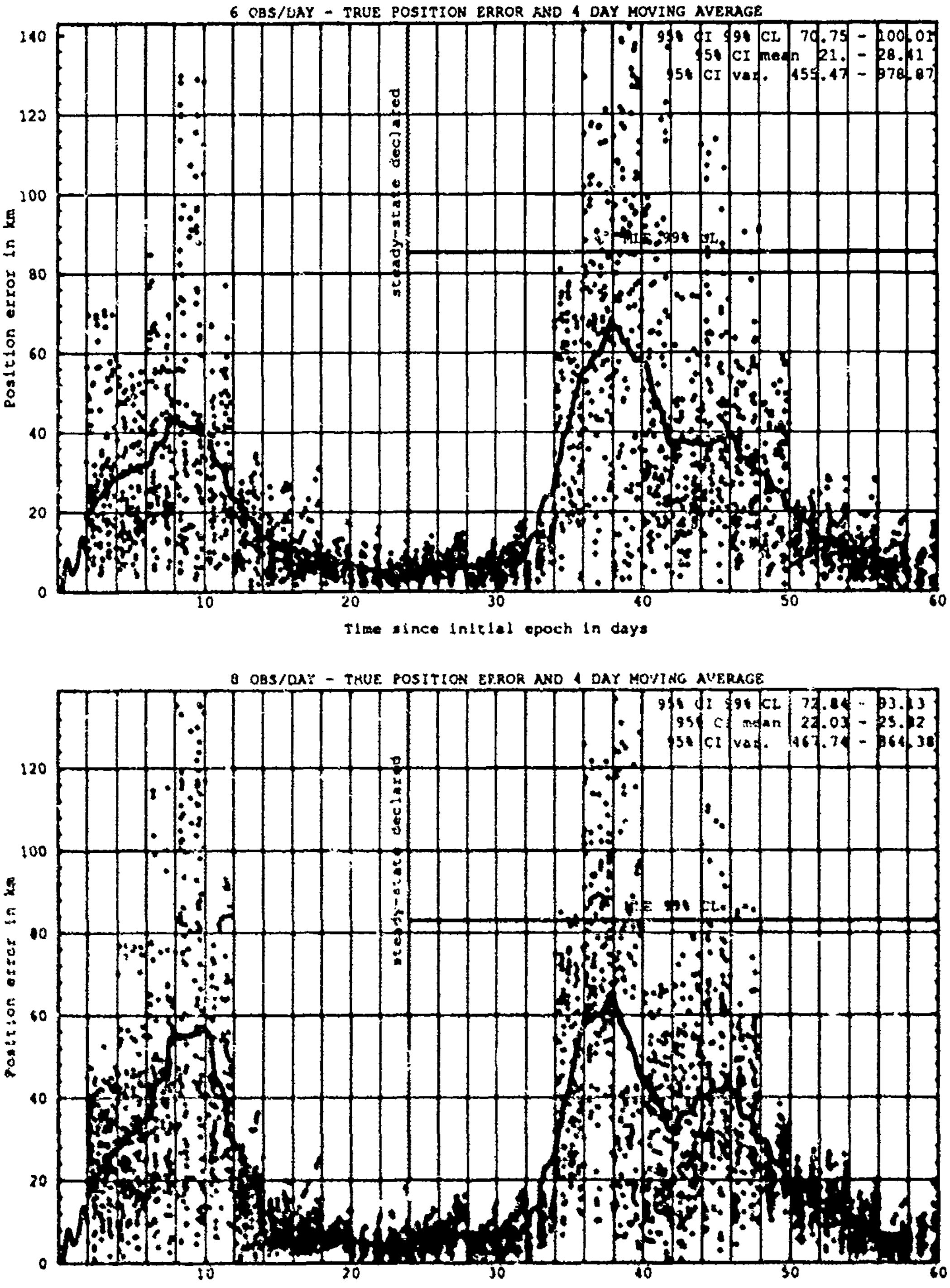
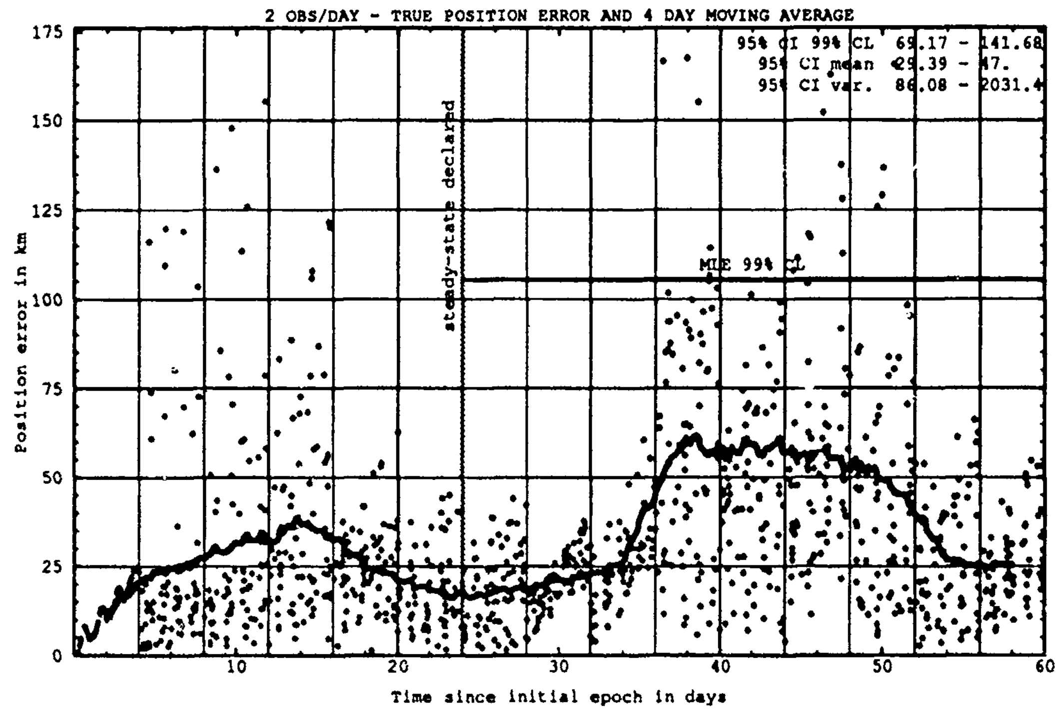


Figure II.2. Class: 1-1-1 (Catalog Number 15141), LUPI 2, OPD 6 and 8.



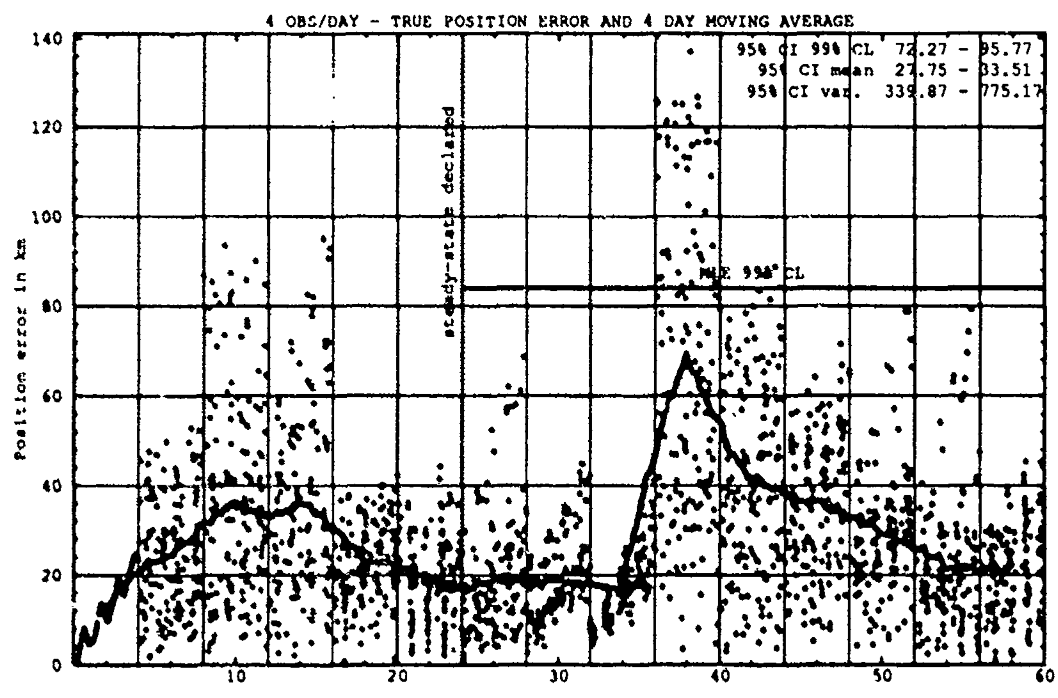
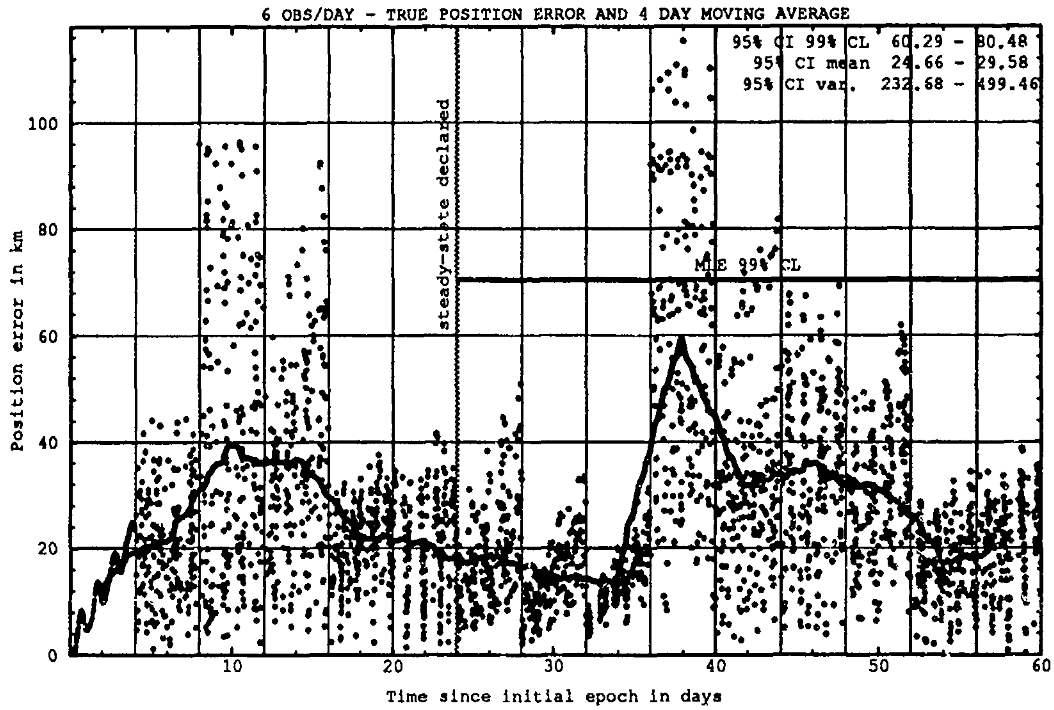


Figure H.3. Class: 1-1-1 (Catalog Number 15141), LUPI 4, OPD 2 and 4.



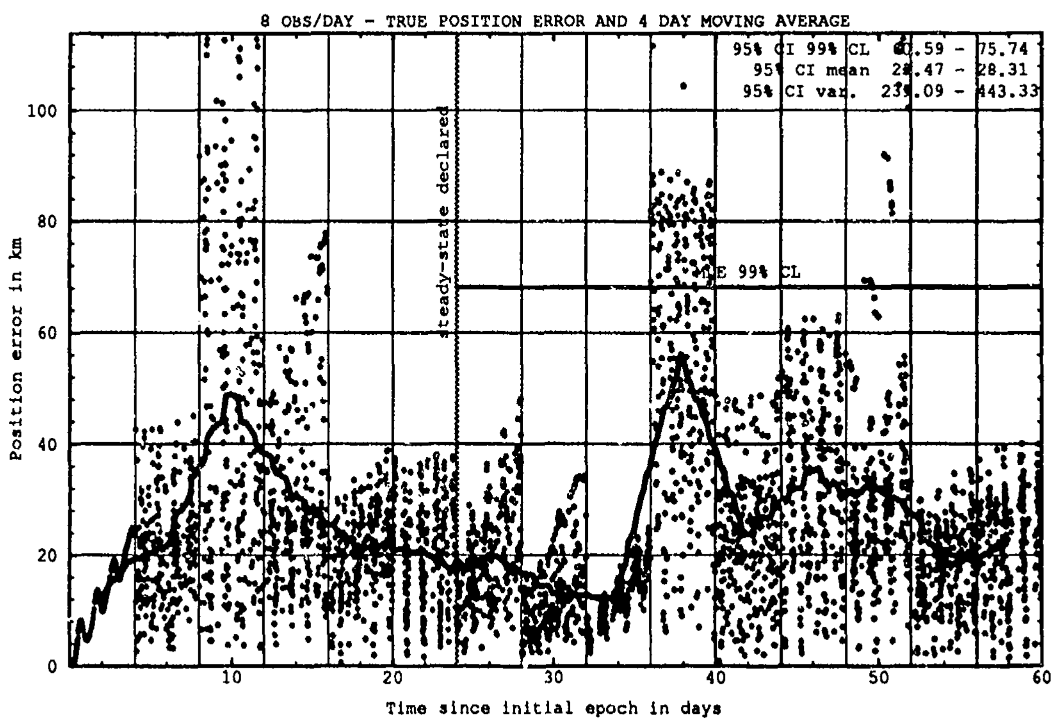
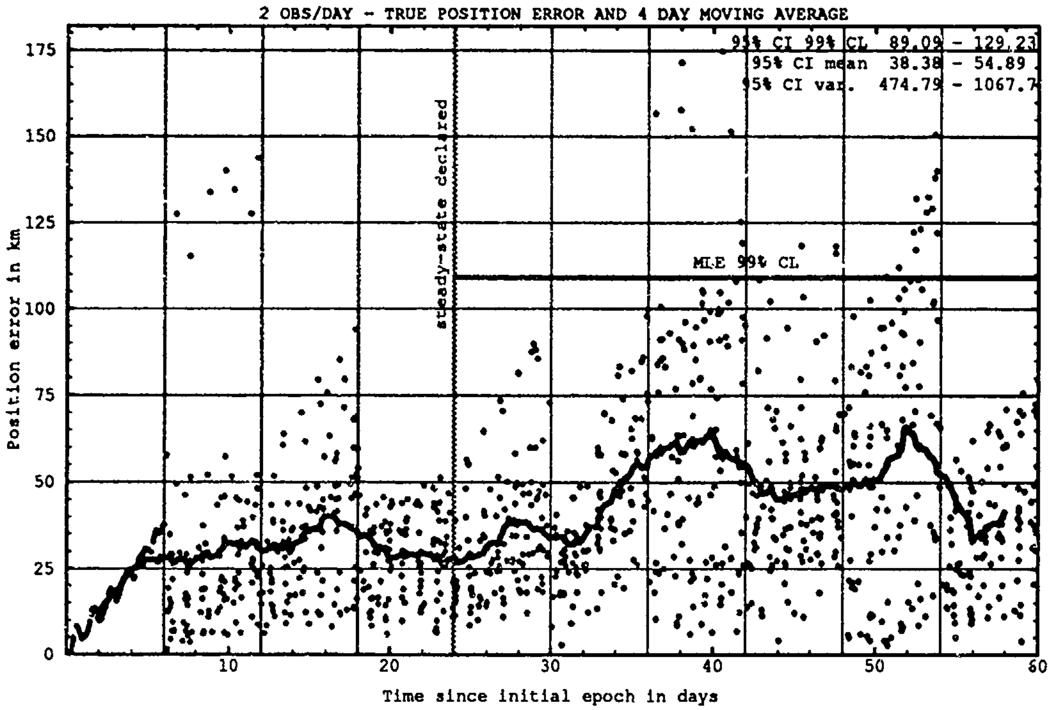


Figure H.4. Class: 1-1-1 (Catalog Number 15141), LUPI 4, OPD 6 and 8.



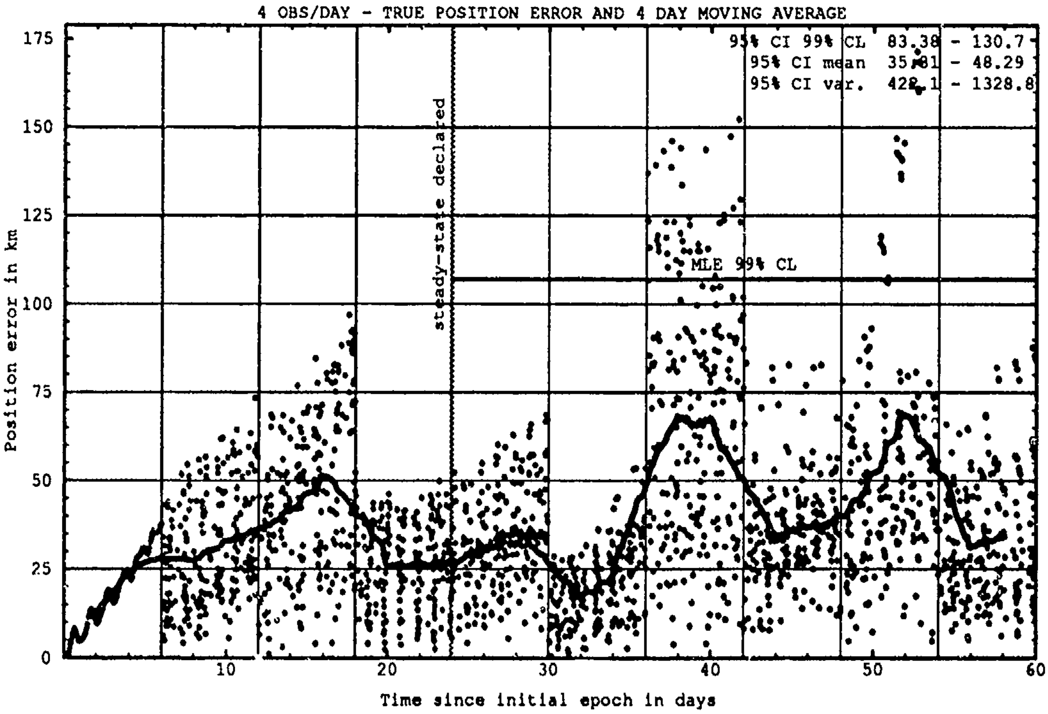
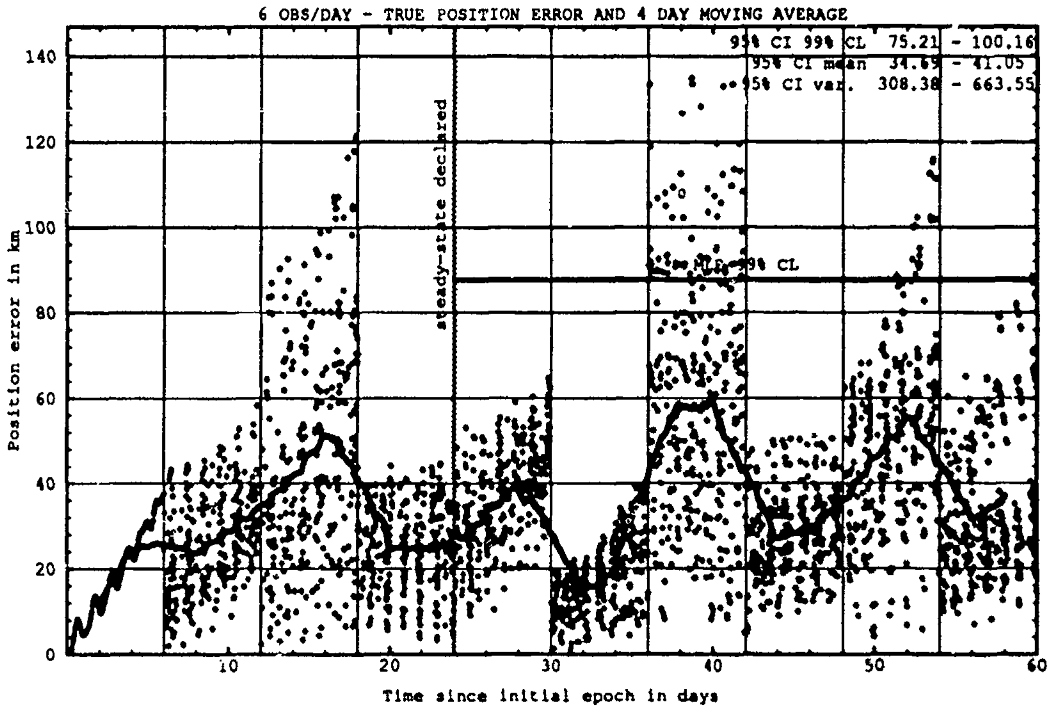


Figure H.5. Class: 1-1-1 (Catalog Number 15141), LUPI 6, OPD 2 and 4.



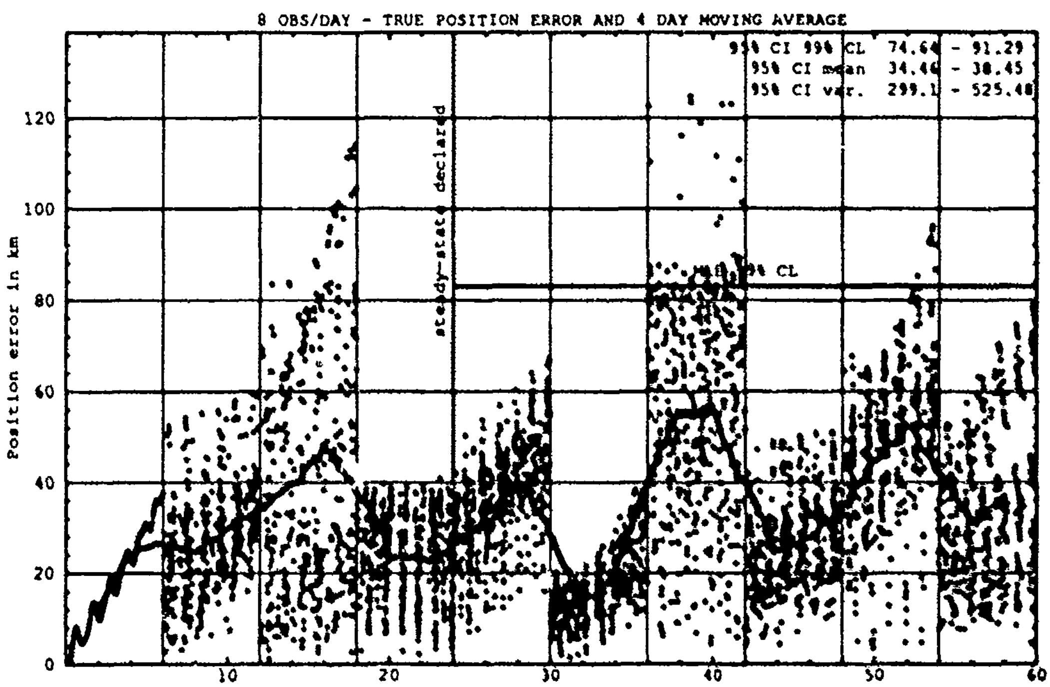
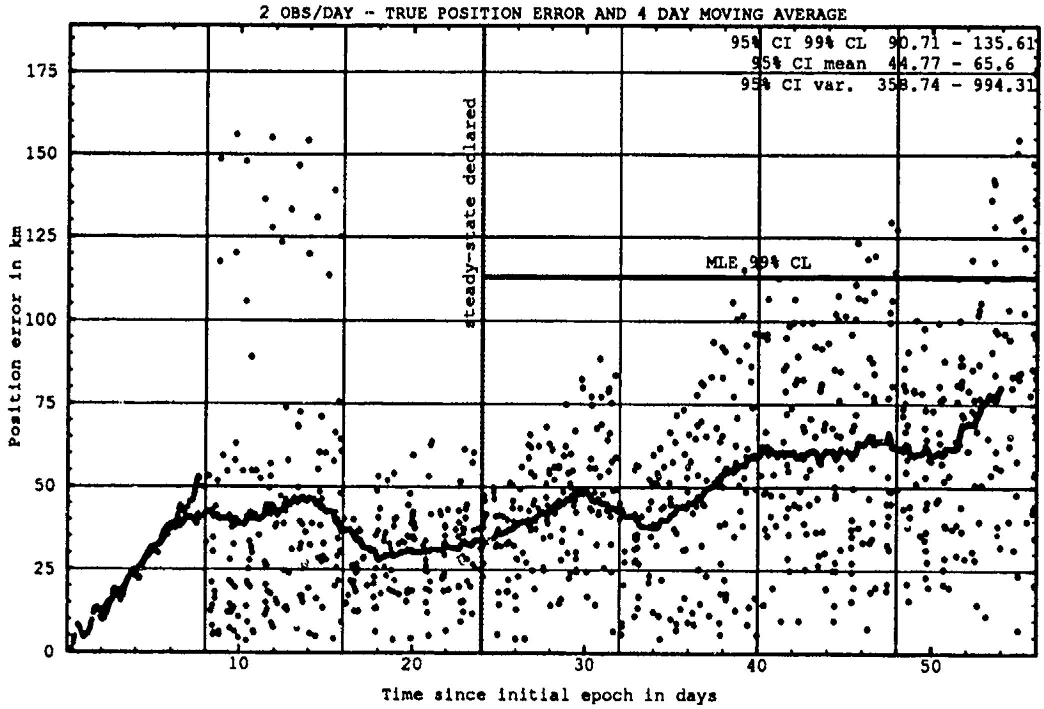


Figure H.6. Class: 1-1-1 (Catalog Number 15141), LUPI 6, OPD 6 and 8.



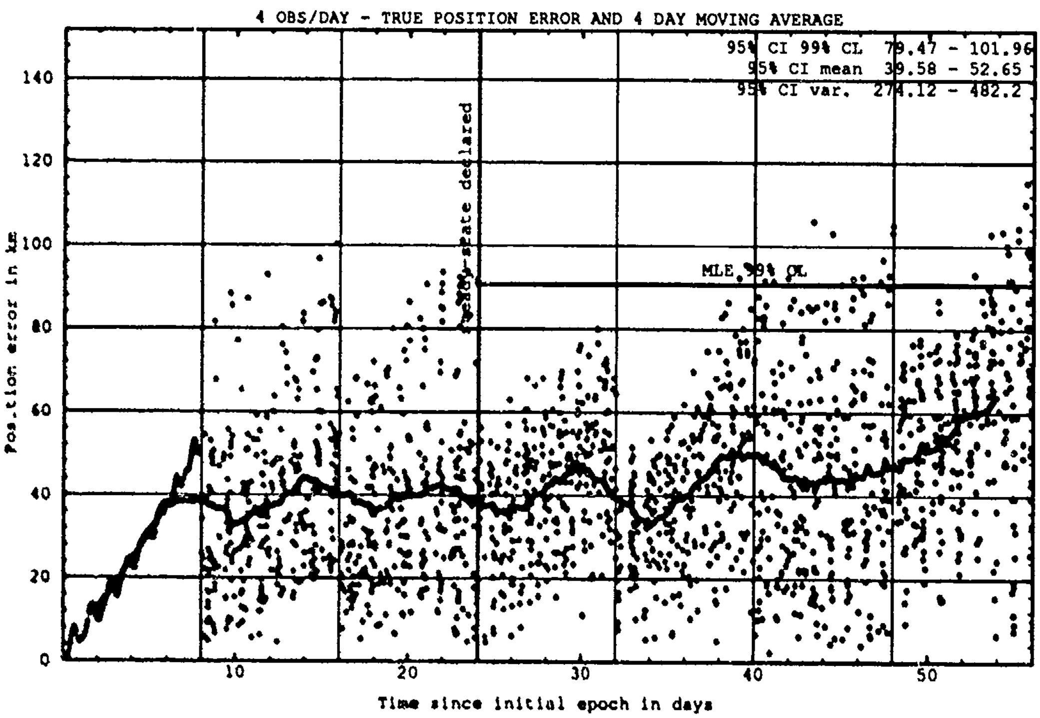


Figure H.7. Class: 1-1-1 (Catalog Number 15141), LUPI 8, OPD 2 and 4.

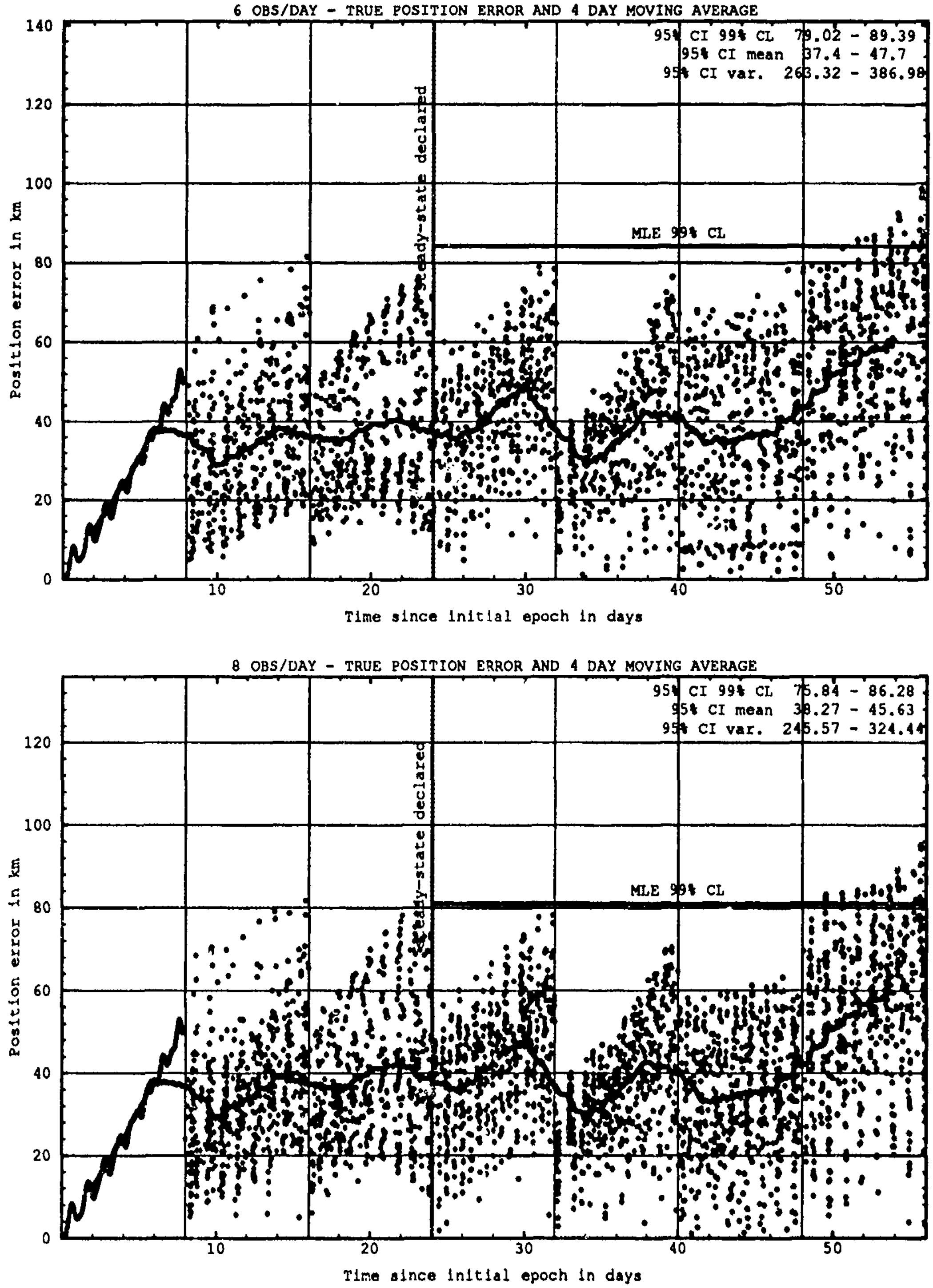


Figure H.8. Class: 1-1-1 (Catalog Number 15141), LUPI 8, OPD 6 and 8.

H.1.2 Differential Corrector Performance. The table below provides statistical information on the last-pass residuals of the differential corrector.

Table H.2. Selected Last-Pass 95% Confidence Interval Statistics.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
15141	8	8	27.02	32.23	231.70	344.92	62.49	75.13

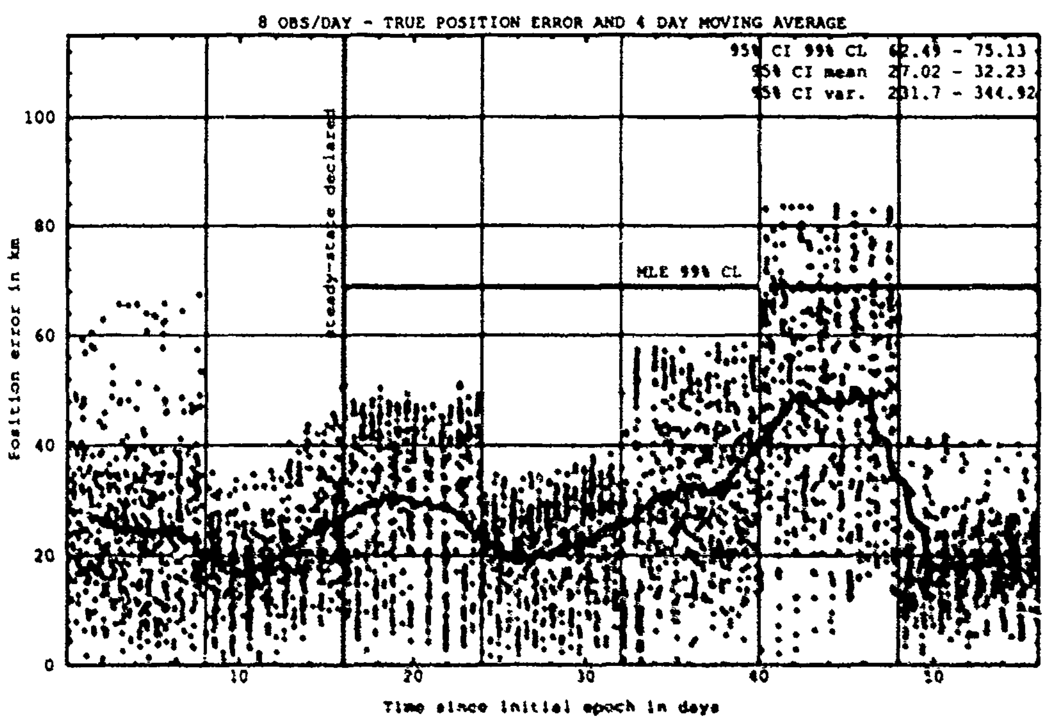


Figure H.9. Last-Pass — Class: 1-1-1 (Catalog Number 15141), LUPI 8, OPD 8.

H.2 CLASS: 1-1-2 (NORAD Catalog Number 15259)

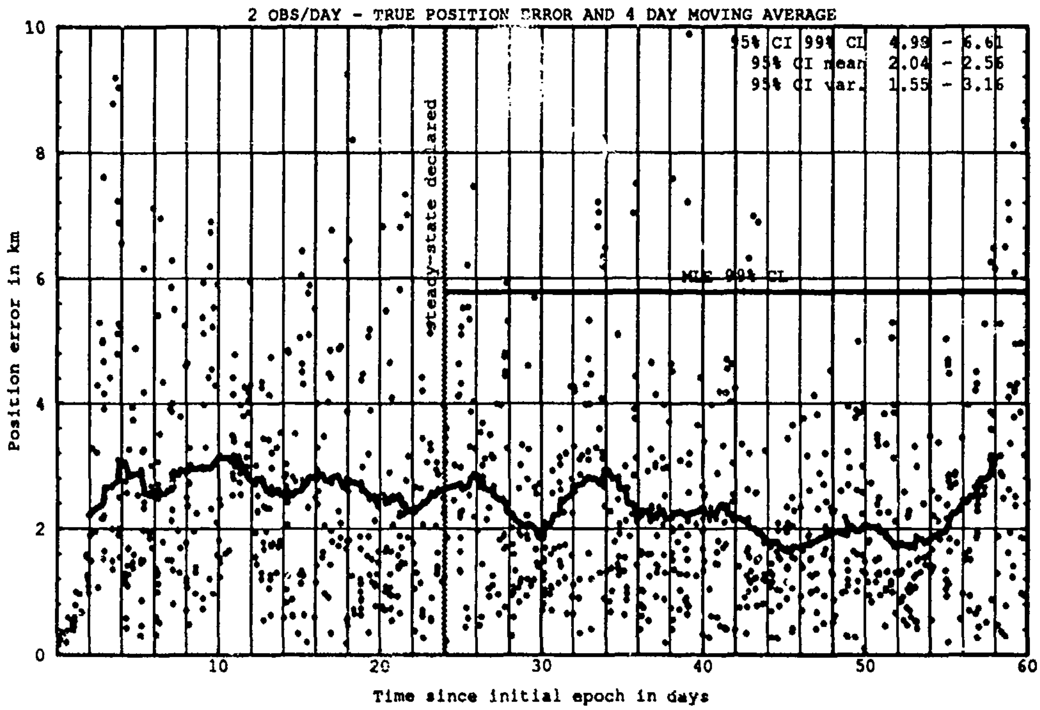
H.2.1 Prediction Performance. The tables below provide the statistical information on the ability of the estimates to predict for 1 LUPI after differential correction.

Table H.4. 95% Confidence Interval Analysis on Class: 1–1–2.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
15259	2	2	2.04	2.56	1.55	3.16	4.98	6.61
15259	2	4	1.83	2.56	1.64	5.20	4.87	7.66
15259	2	6	1.61	2.28	1.14	3.82	4.22	6.64
15259	2	8	1.52	2.00	1.33	3.57	4.19	6.29
15259	4	2	2.16	2.94	1.52	7.42	5.31	8.86
15259	4	4	1.98	3.06	2.24	7.77	5.40	9.34
15259	4	6	1.43	2.09	0.85	3.52	3.60	6.28
15259	4	8	1.03	1.46	0.42	1.34	2.59	4.03
15259	6	2	2.18	3.54	1.90	8.56	5.57	9.96
15259	6	4	1.28	2.14	0.36	3.75	3.17	6.26
15259	6	6	1.16	1.67	0.35	1.83	2.69	4.63
15259	6	8	0.91	1.24	0.24	0.81	2.15	3.21
15259	8	2	2.22	3.89	2.26	10.97	5.64	11.28
15259	8	4	1.38	1.86	1.05	2.20	3.82	5.18
15259	8	6	1.36	1.89	0.97	2.07	3.75	5.11
15259	8	8	1.16	1.80	0.64	2.15	3.12	5.06

Table H.5. ANOVA Analysis on Class: 1–1–2.

Source of Variation	Degrees of Freedom	Sum of Square	Mean Square	F	Table F
Replications	9	84.46	9.38	2.92	1.88
Main Effects:					
LUPI	3	25.34	8.45	2.62	2.60
OPD	3	267.94	89.31	27.75	2.60
Interaction	9	122.42	13.60	4.23	1.88
Error	135	434.57	3.22		
Total	159	934.72			



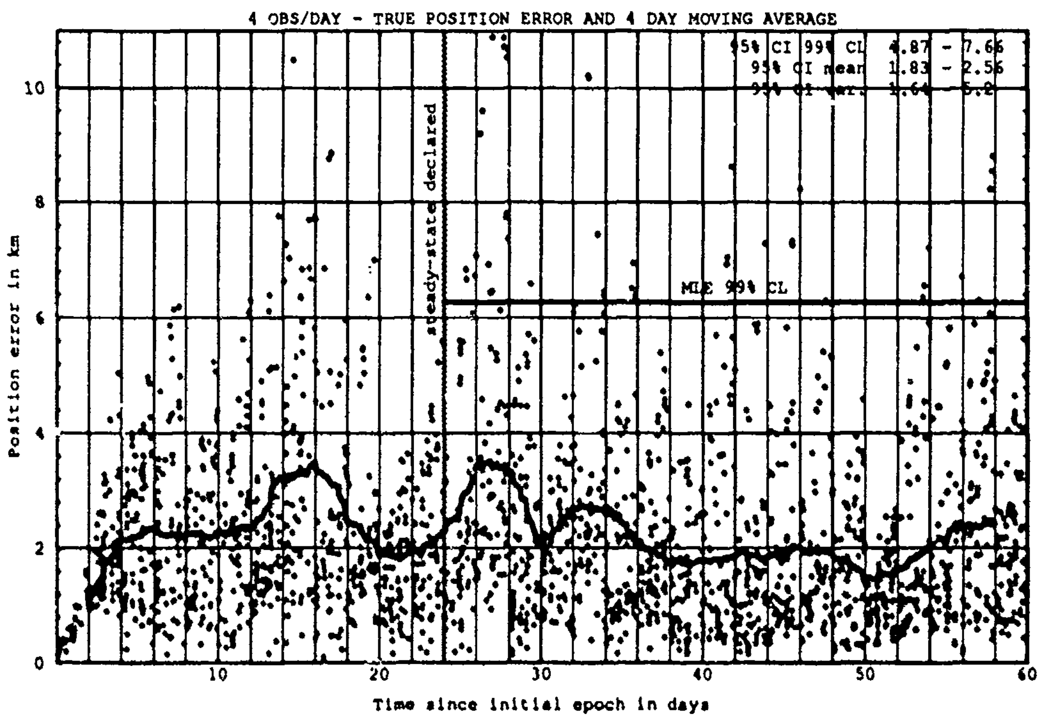
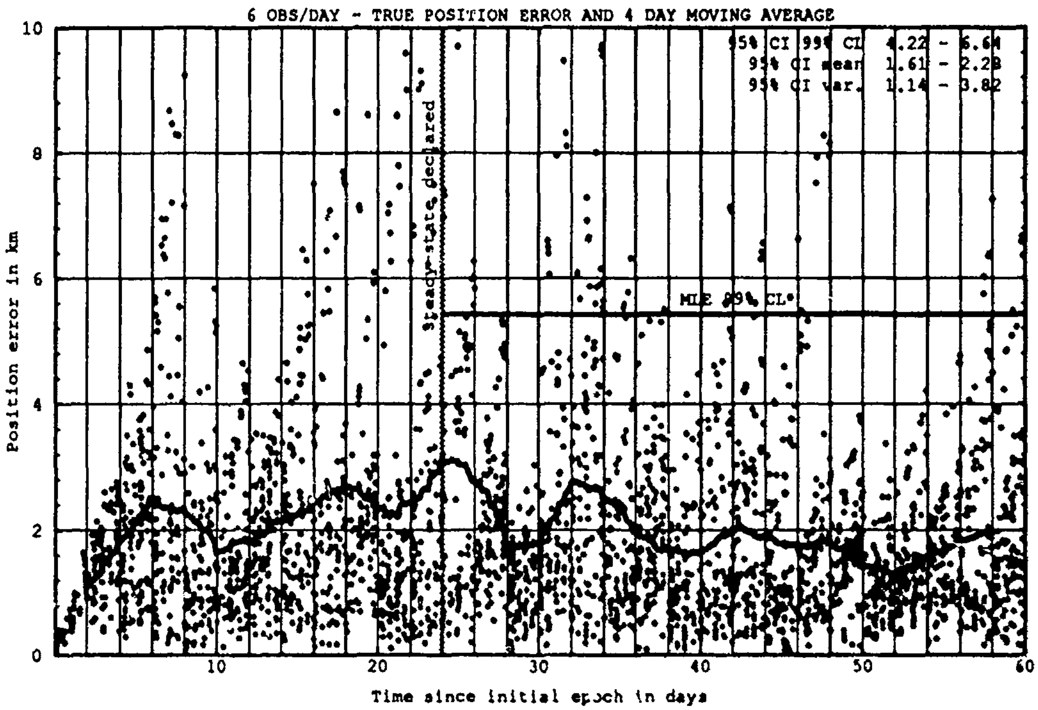


Figure H.10. Class: 1-1-2 (Catalog Number 15259), LUPI 2, OPD 2 and 4.



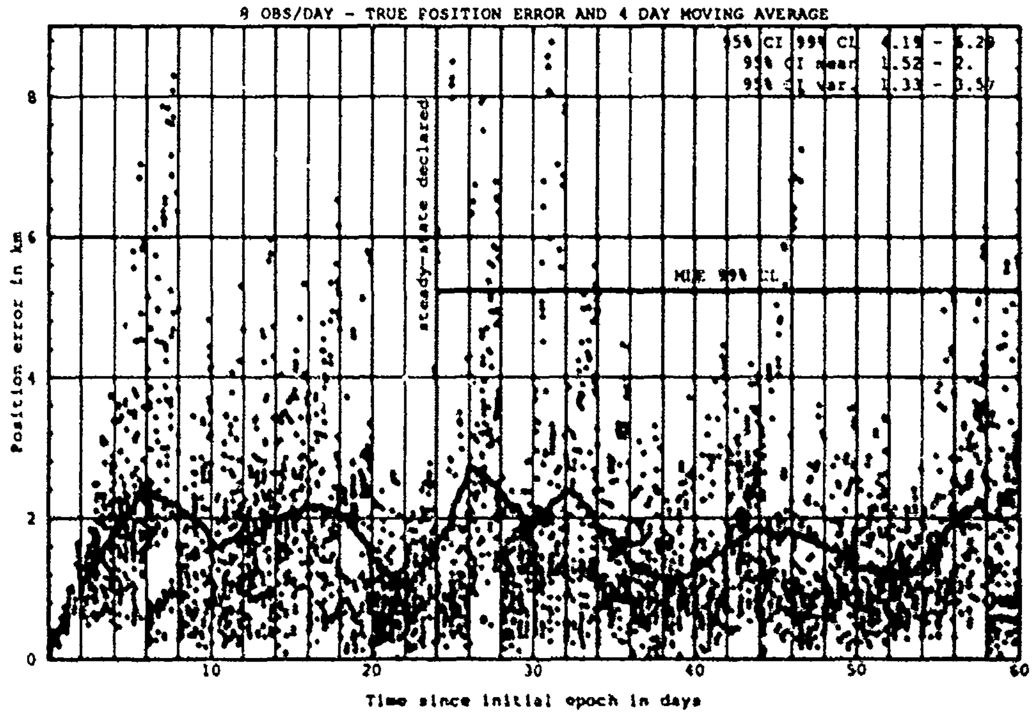
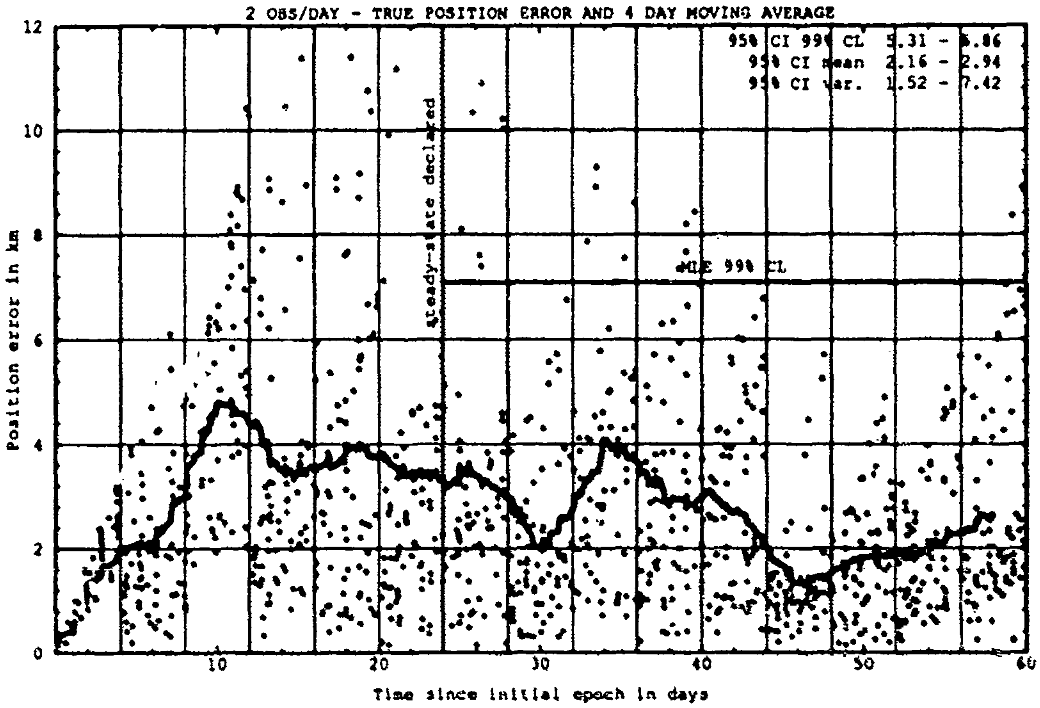


Figure H.11. Class: 1-1-2 (Catalog Number 15259), LUPI 2, OPD 6 and 8.



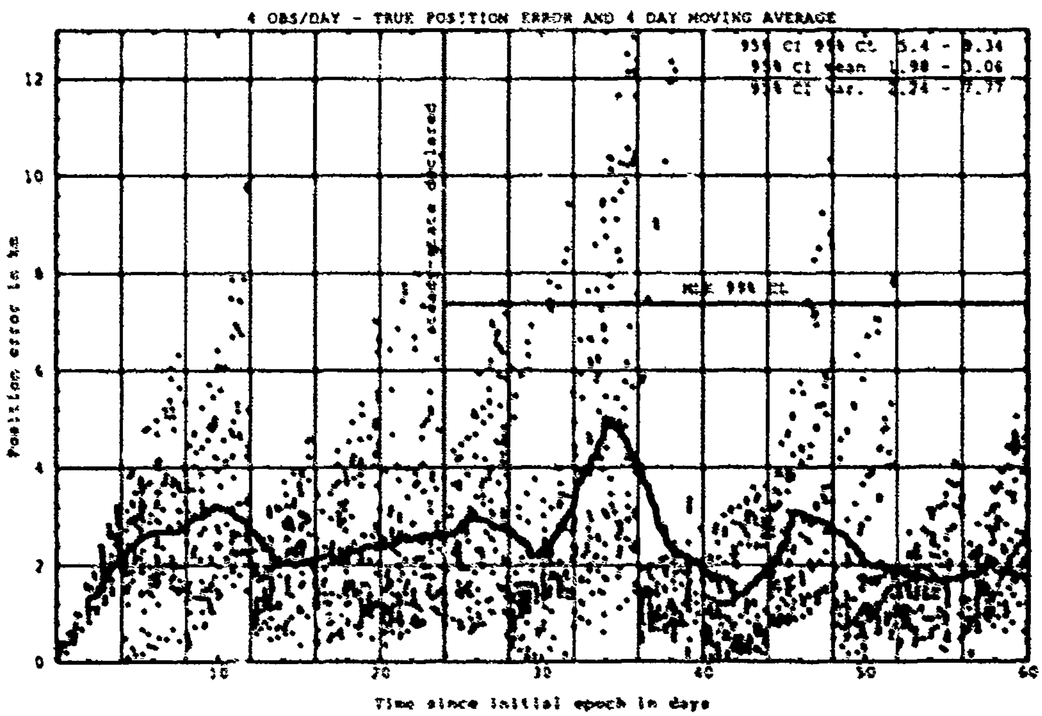
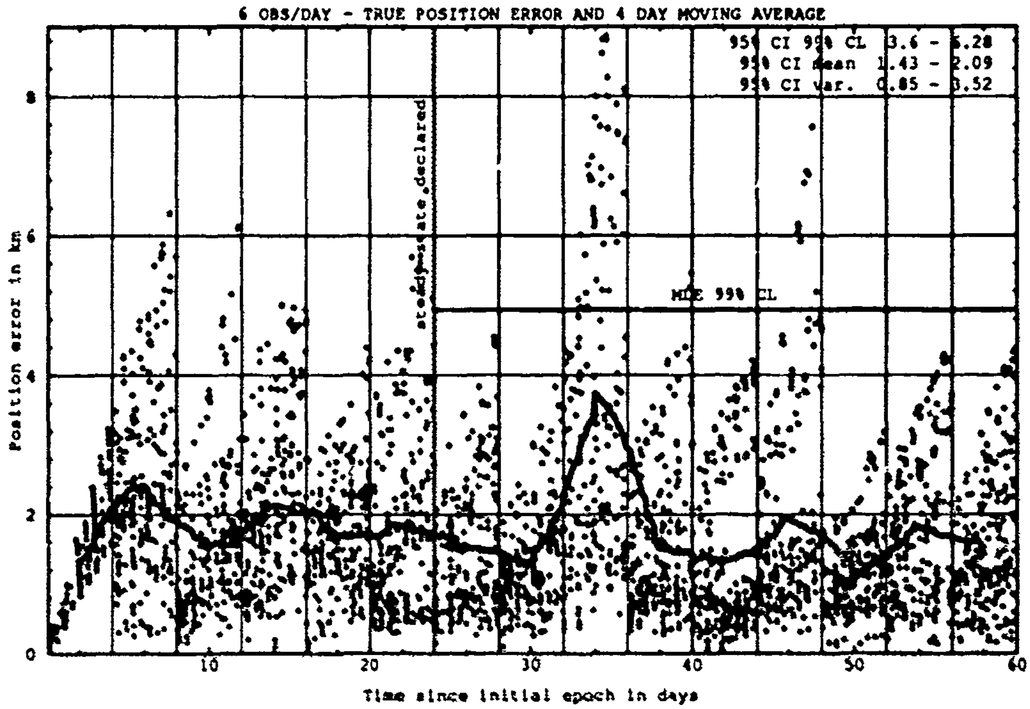


Figure H.12. Class: 1-1-2 (Catalog Number 15259), LUPI 4, OPD 2 and 4.



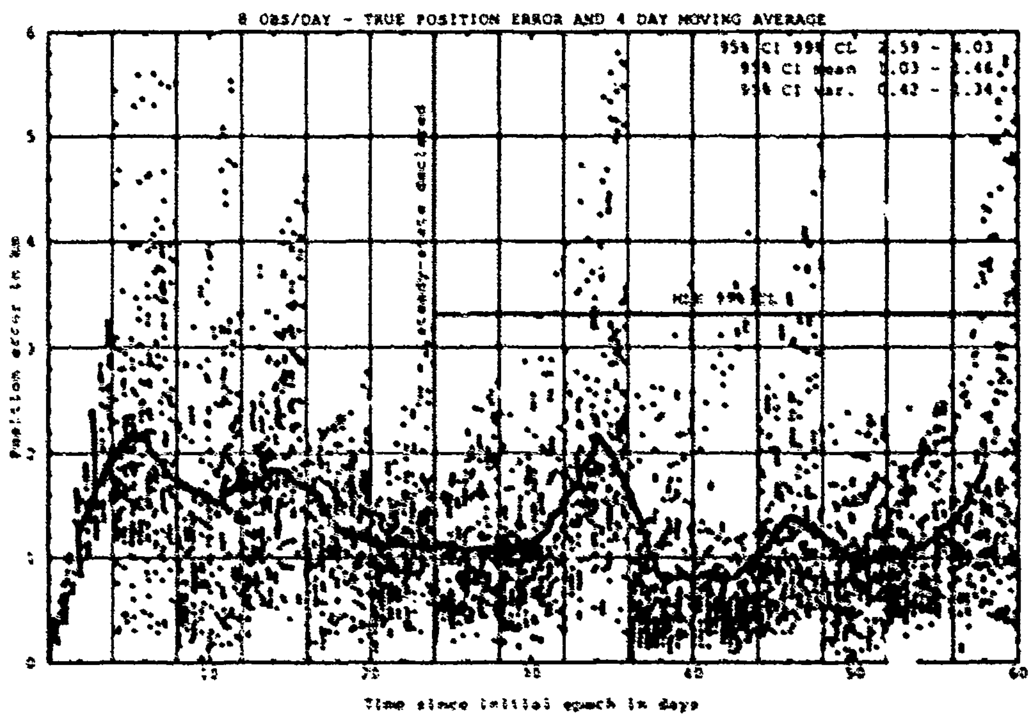
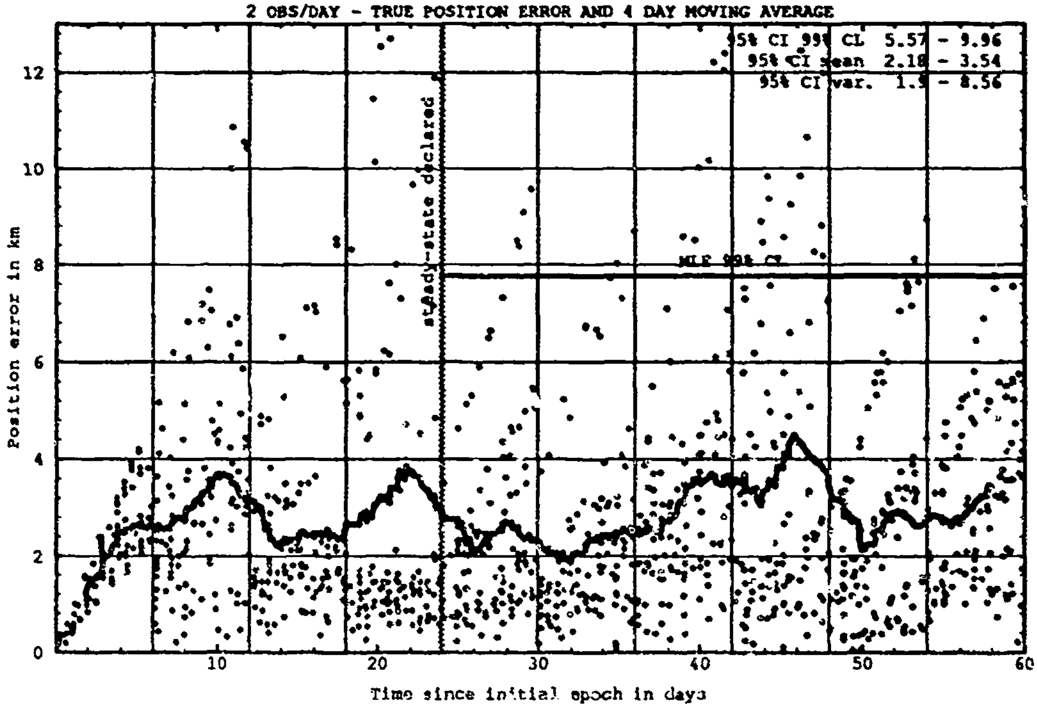


Figure H.13. Class: 1-1-2 (Catalog Number 15259), LUPI 4, OPD 6 and 8.



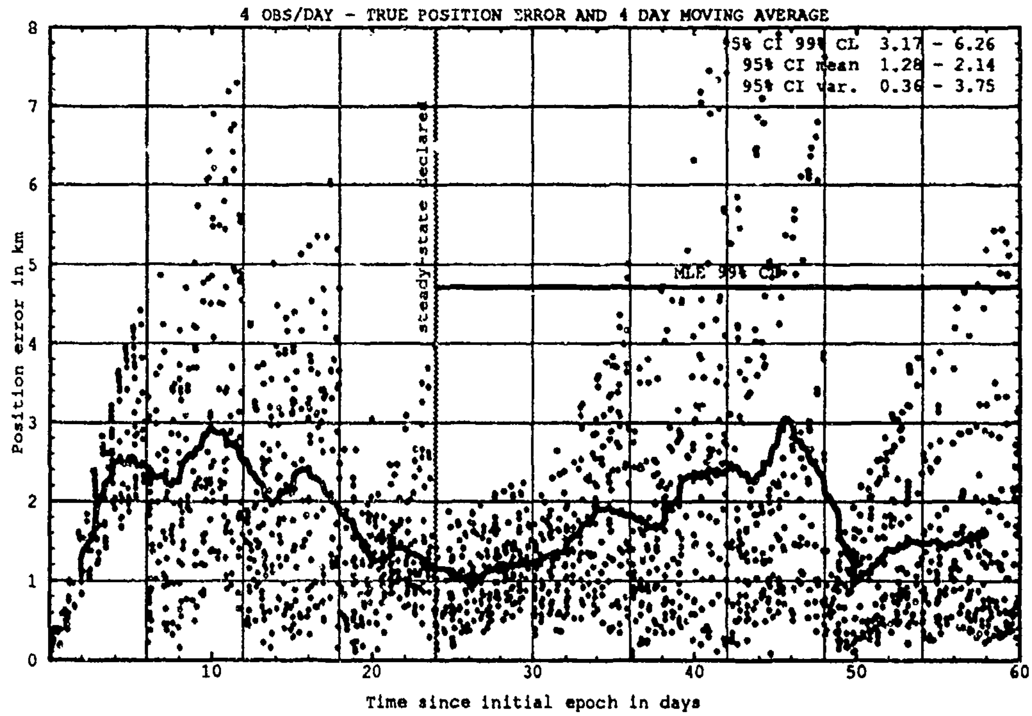
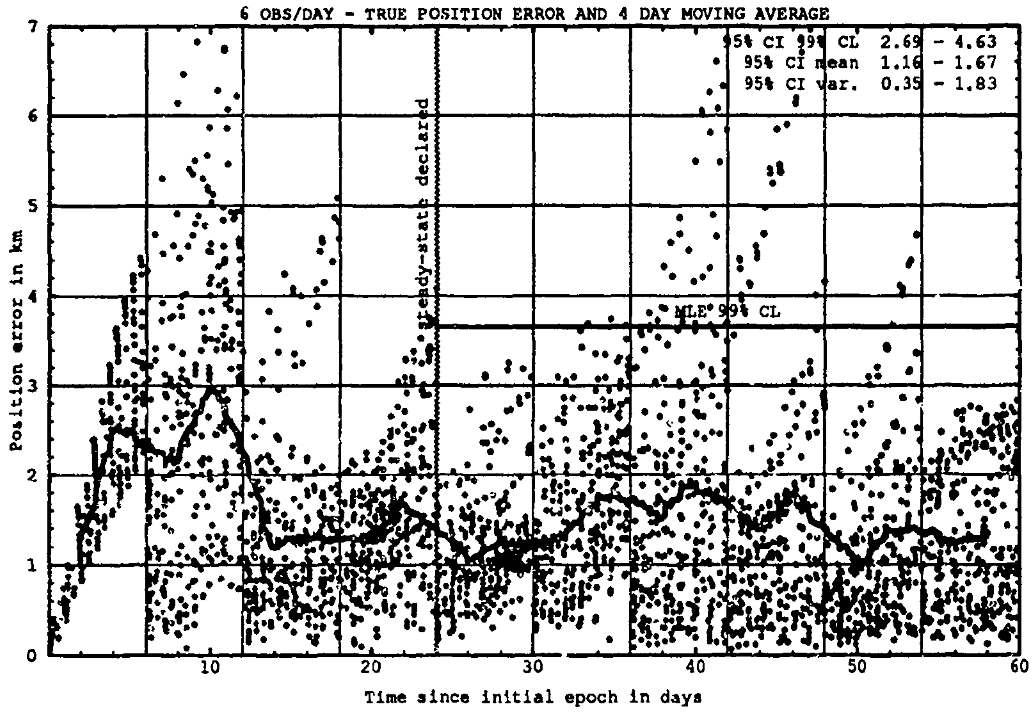


Figure H.14. Class: 1-1-2 (Catalog Number 15259), LUPI 6, OPD 2 and 4.



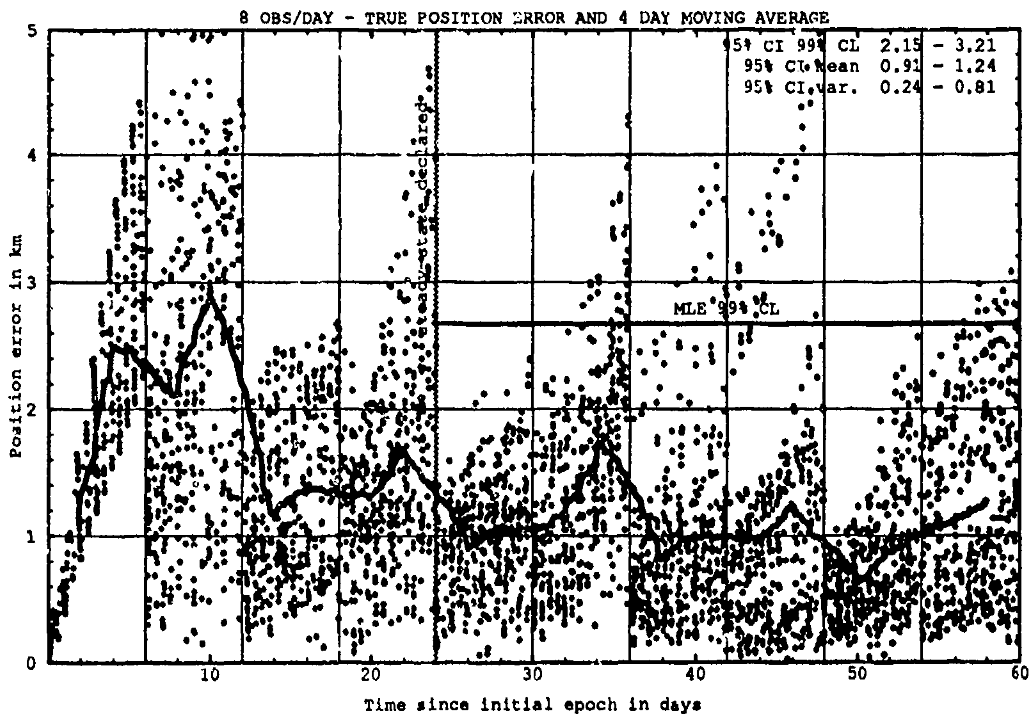


Figure H.15. Class: 1-1-2 (Catalog Number 15259), LUPI 6, OPD 6 and 8.

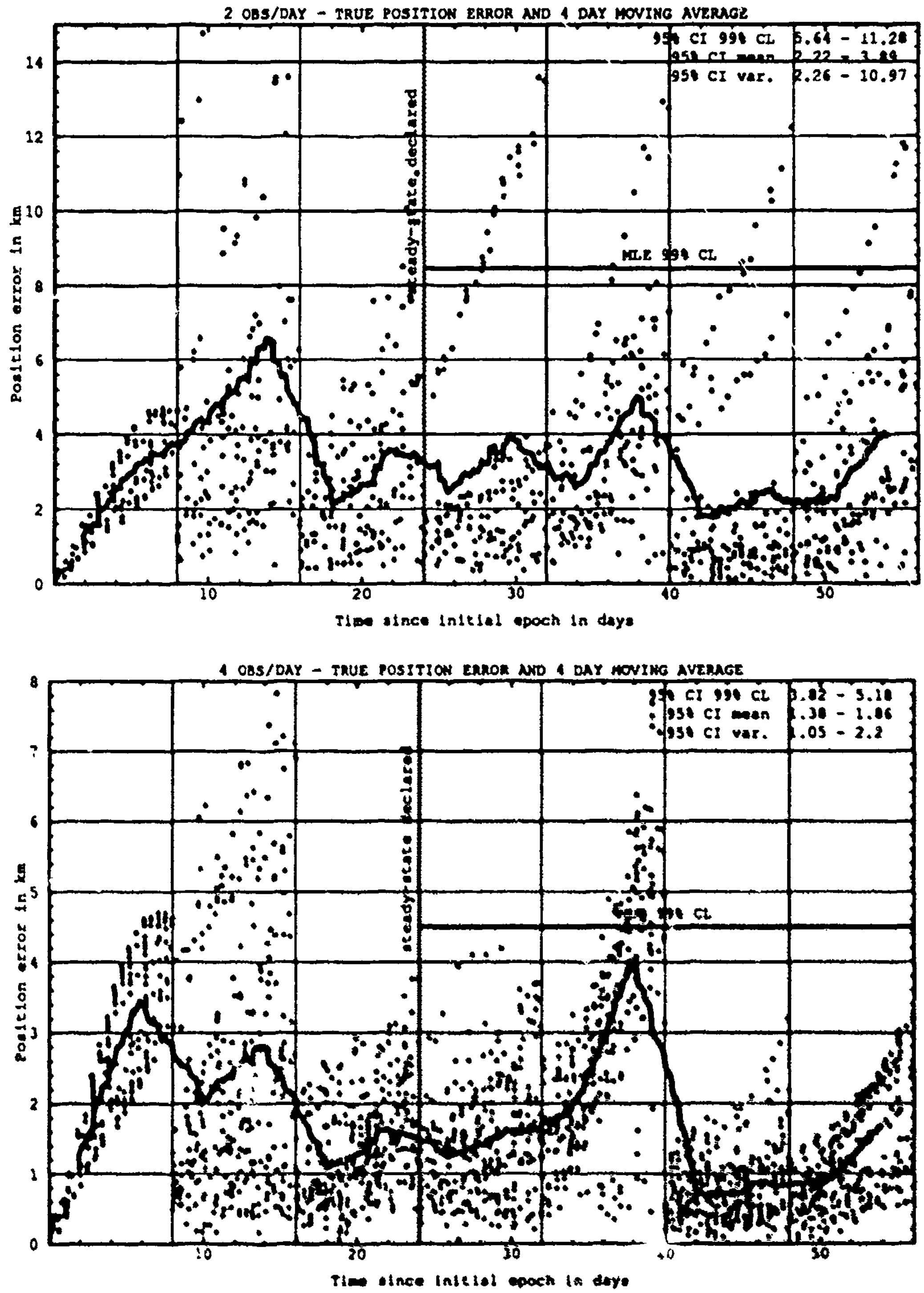
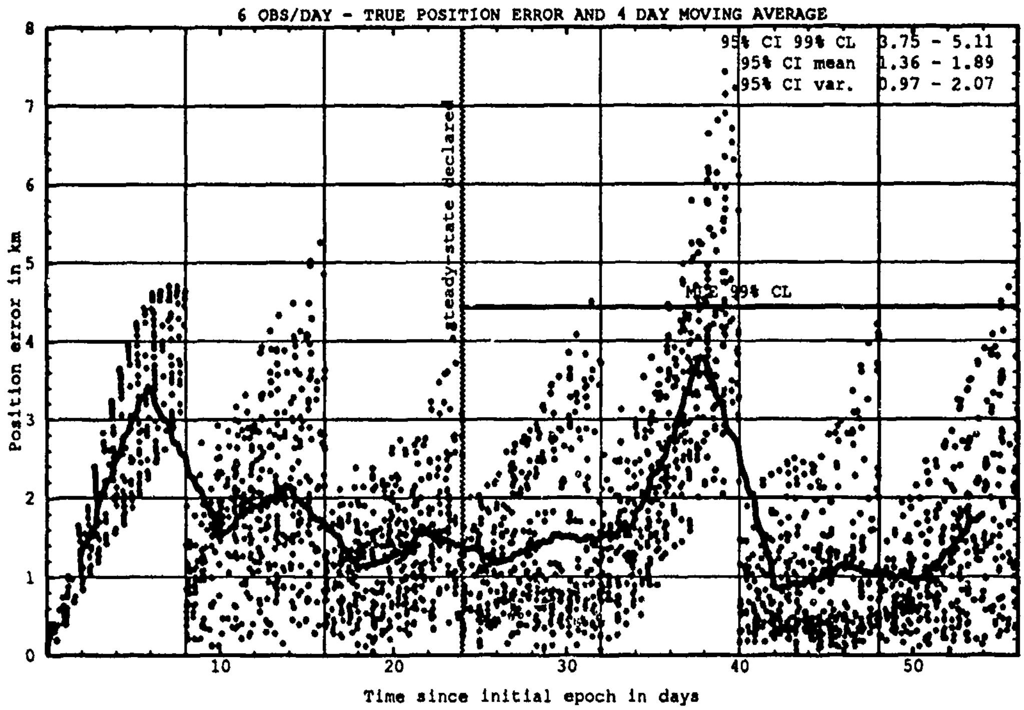


Figure H.16. Class: 1-1-2 (Catalog Number 15259), LUPI 8, OPD 2 and 4.



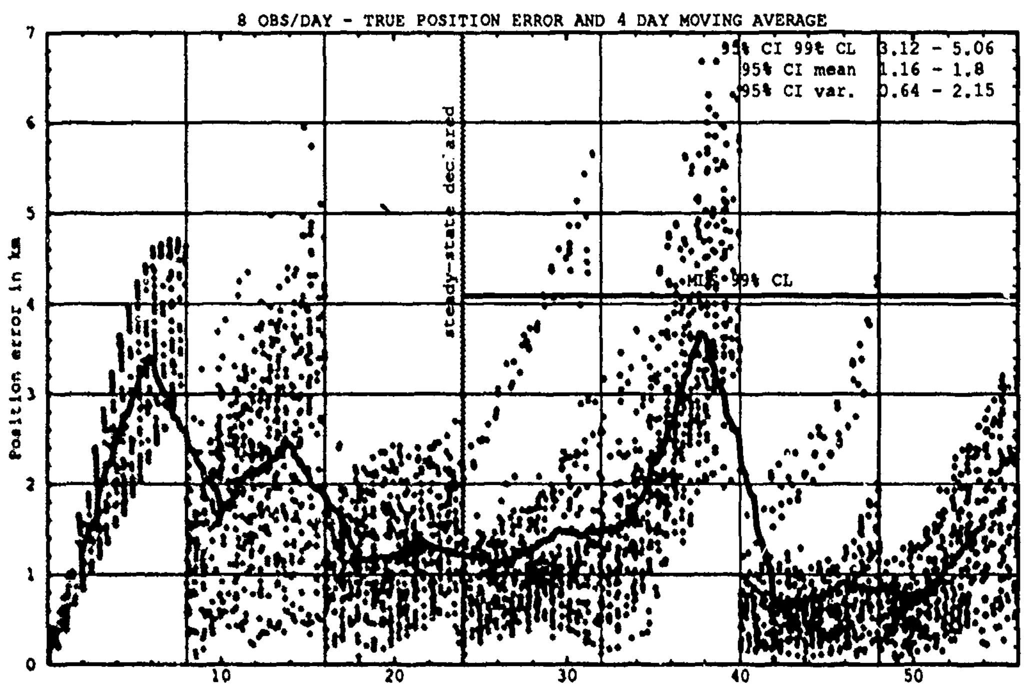


Figure H.17. Class: 1-1-2 (Catalog Number 15259), LUPI 8, OPD 6 and 8.

H.2.2 Differential Corrector Performance. The table below provides statistical information on the last-pass residuals of the differential corrector.

Table H.4. Selected Last-Pass 95% Confidence Interval Statistics.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
15259	8	8	0.57	0.75	0.10	0.17	1.30	1.70

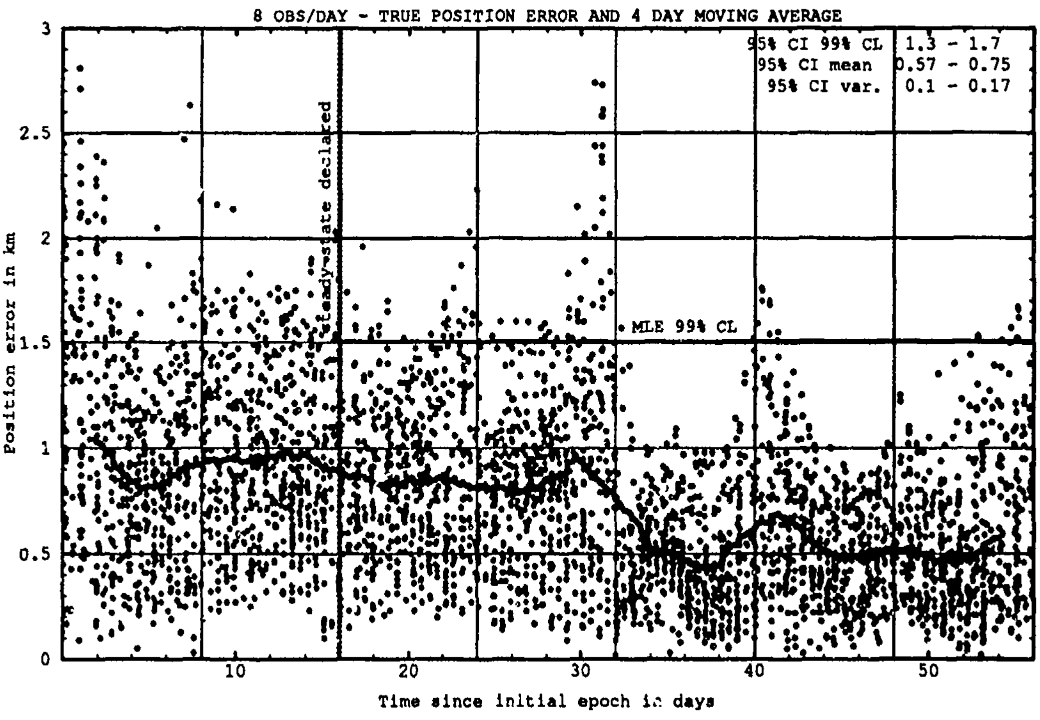


Figure H.18. Last-Pass — Class: 1–1–2 (Catalog Number 15259), LUPI 8, OPD 8.

H.3 CLASS: 1-3-2 (NORAD Catalog Number 14199)

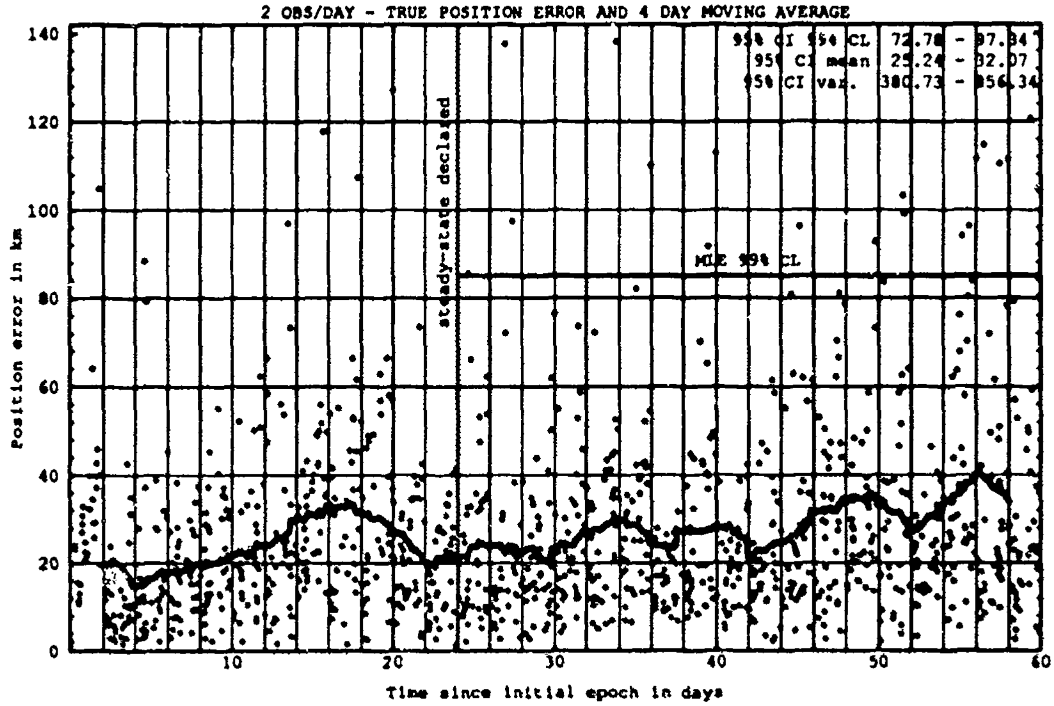
H.3.1 Prediction Performance. The tables below provide the statistical information on the ability of the estimates to predict for 1 LUPI after differential correction.

Table H.7. 95% Confidence Interval Analysis on Class: 1–3–2.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
14199	2	2	25.24	32.07	380.73	856.34	72.78	97.34
14199	2	4	22.33	28.87	263.28	671.48	58.85	88.53
14199	2	6	16.01	20.54	78.17	387.41	40.13	63.22
14199	2	8	11.89	14.71	70.56	119.53	32.08	39.29
14199	4	2	35.98	49.00	665.86	1630.12	95.68	141.01
14199	4	4	19.94	27.64	56.91	845.67	49.68	87.73
14199	4	6	13.08	17.92	109.61	203.29	37.69	50.42
14199	4	8	10.47	13.77	60.06	107.34	28.63	37.50
14199	6	2	42.48	63.08	934.39	3612.94	115.09	197.46
14199	6	4	17.14	24.32	77.81	497.16	43.40	71.67
14199	6	6	13.00	16.73	94.78	182.74	36.27	47.30
14199	6	8	11.85	14.73	59.88	122.89	30.26	39.74
14199	8	2	32.15	48.88	459.80	2118.17	85.23	150.42
14199	8	4	16.43	26.71	143.11	450.05	45.54	73.89
14199	8	6	15.09	21.75	132.45	330.04	42.38	62.67
14199	8	8	14.33	18.56	90.98	212.99	36.74	51.51

Table H.8. ANOVA Analysis on Class: 1–3–2.

Source of Variation	Degrees of Freedom	Sum of Square	Mean Square	F	Table F
Replications	9	130[illegible]	1444.46	0.2482	1.88
Main Effects:					
LUPI	3	2601.79	867.26	0.1490	2.60
OPD	3	161167.8	53722.6	9.23	2.60
Interaction	9	26143.48	2904.83	0.4991	1.88
Error	135	785705.8	5820.04		
Total	159	988618.99			



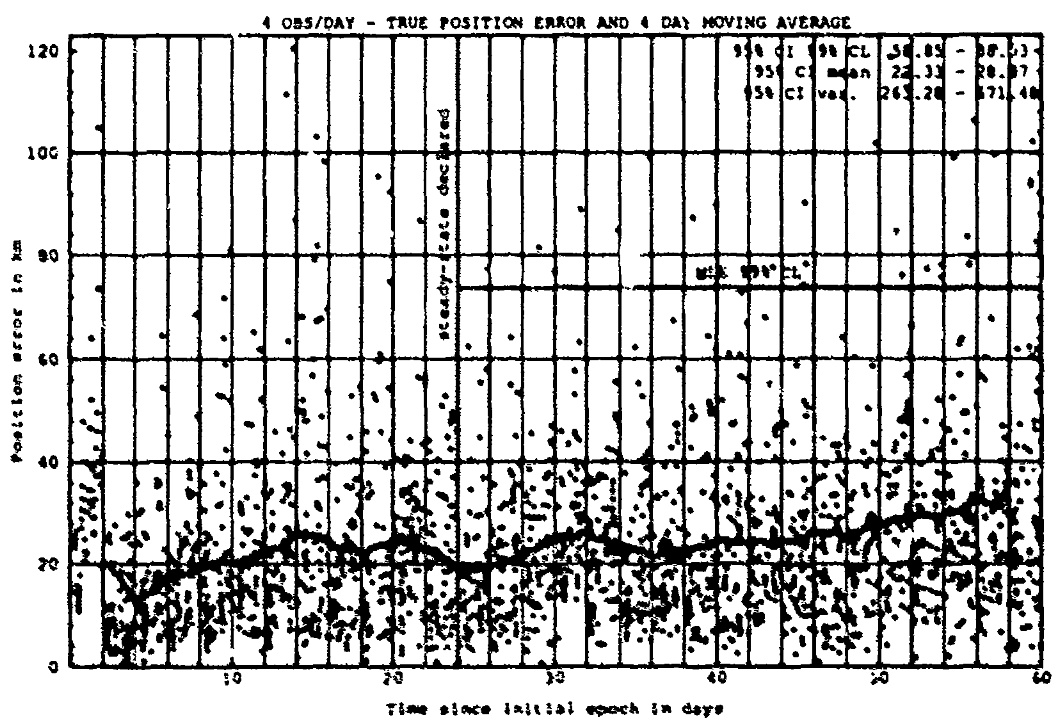
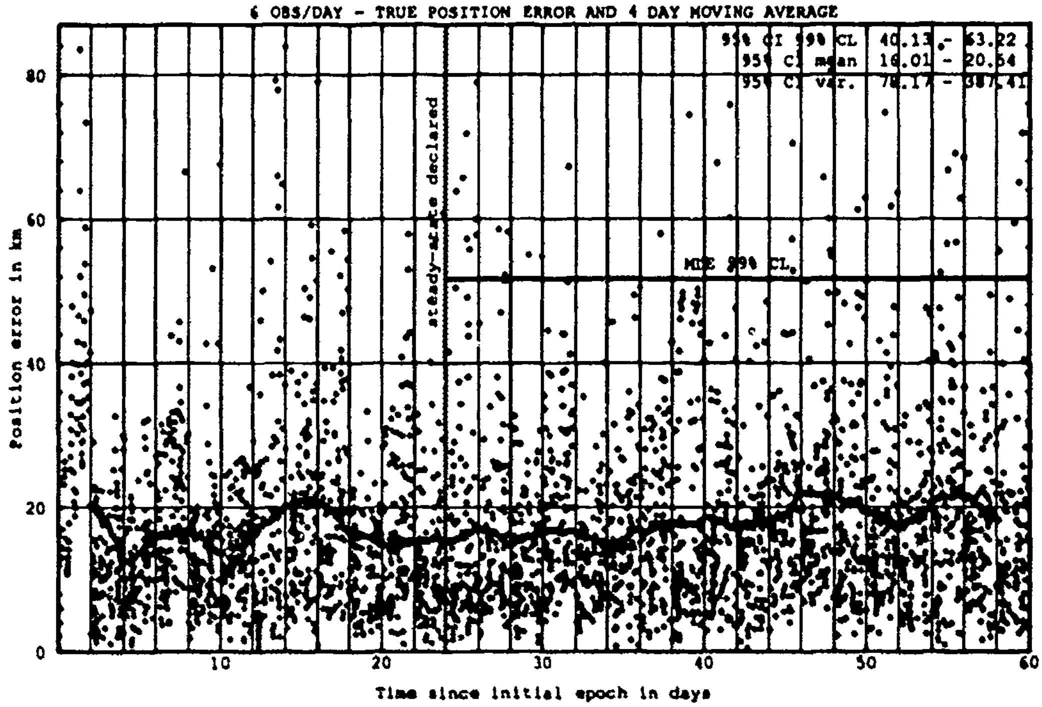


Figure H.19. Class: 1-3-2 (Catalog Number 14199), LUPI 2, OPD 2 and 4.



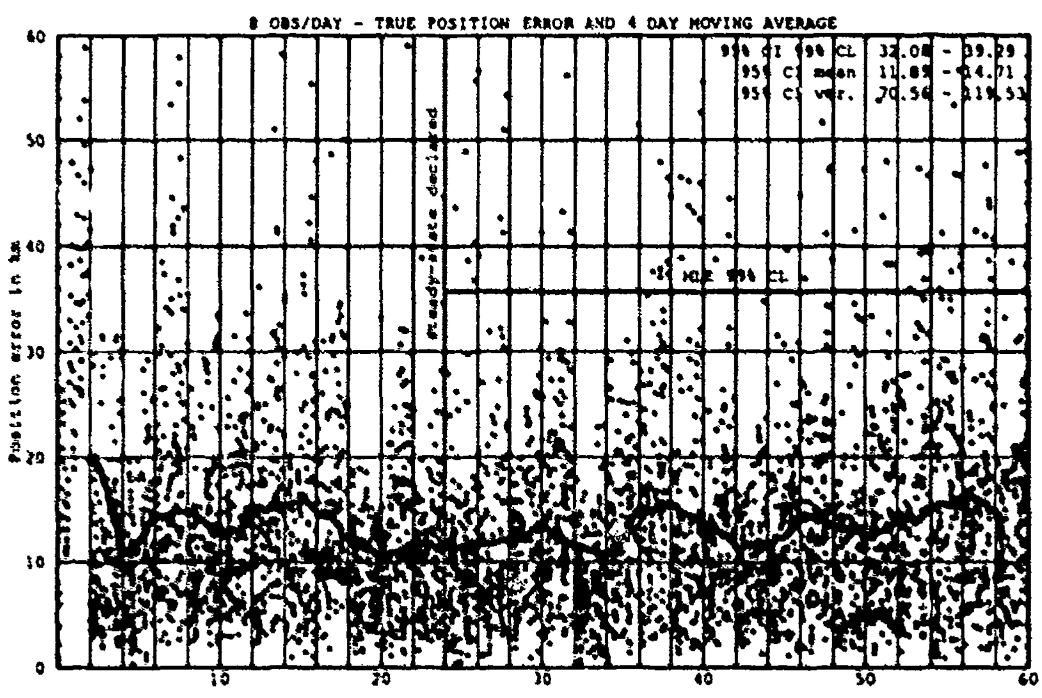
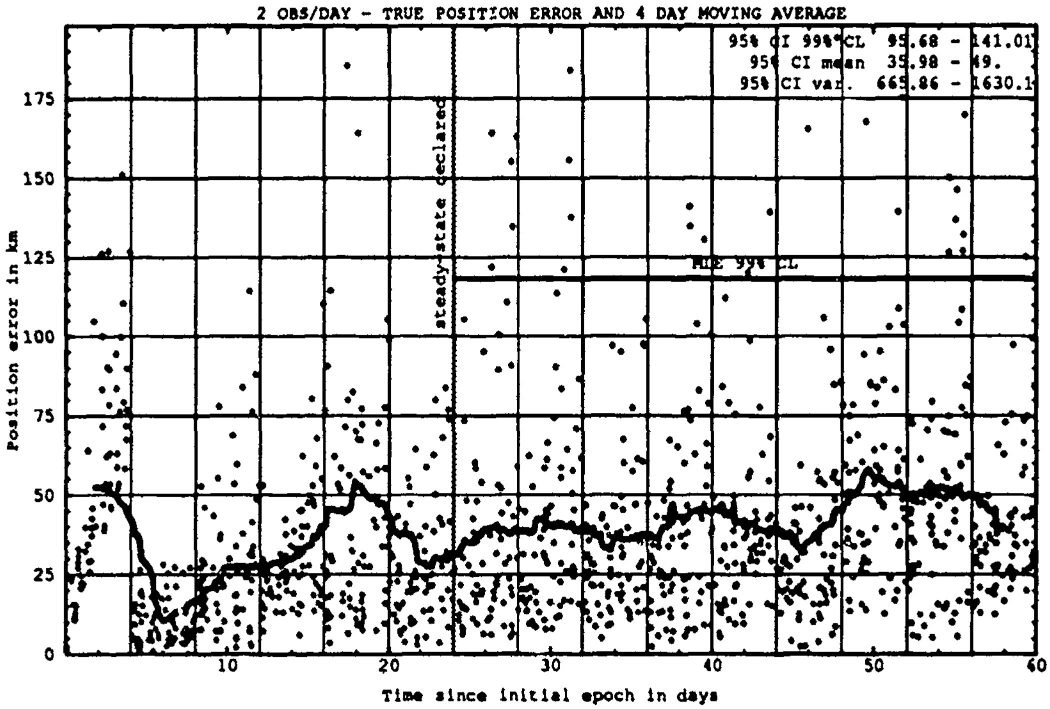


Figure H.20. Class: 1-3-2 (Catalog Number 14199), LUPI 2, OPD 6 and 8.



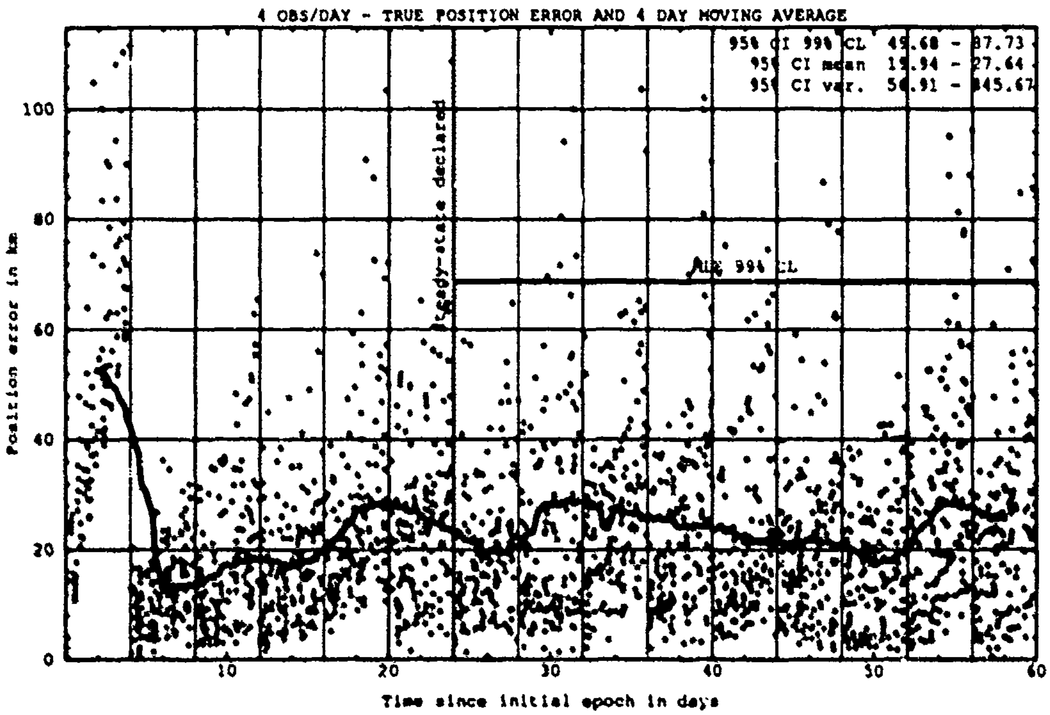


Figure H.21. Class: 1-3-2 (Catalog Number 14199), LUPI 4, OPD 2 and 4.

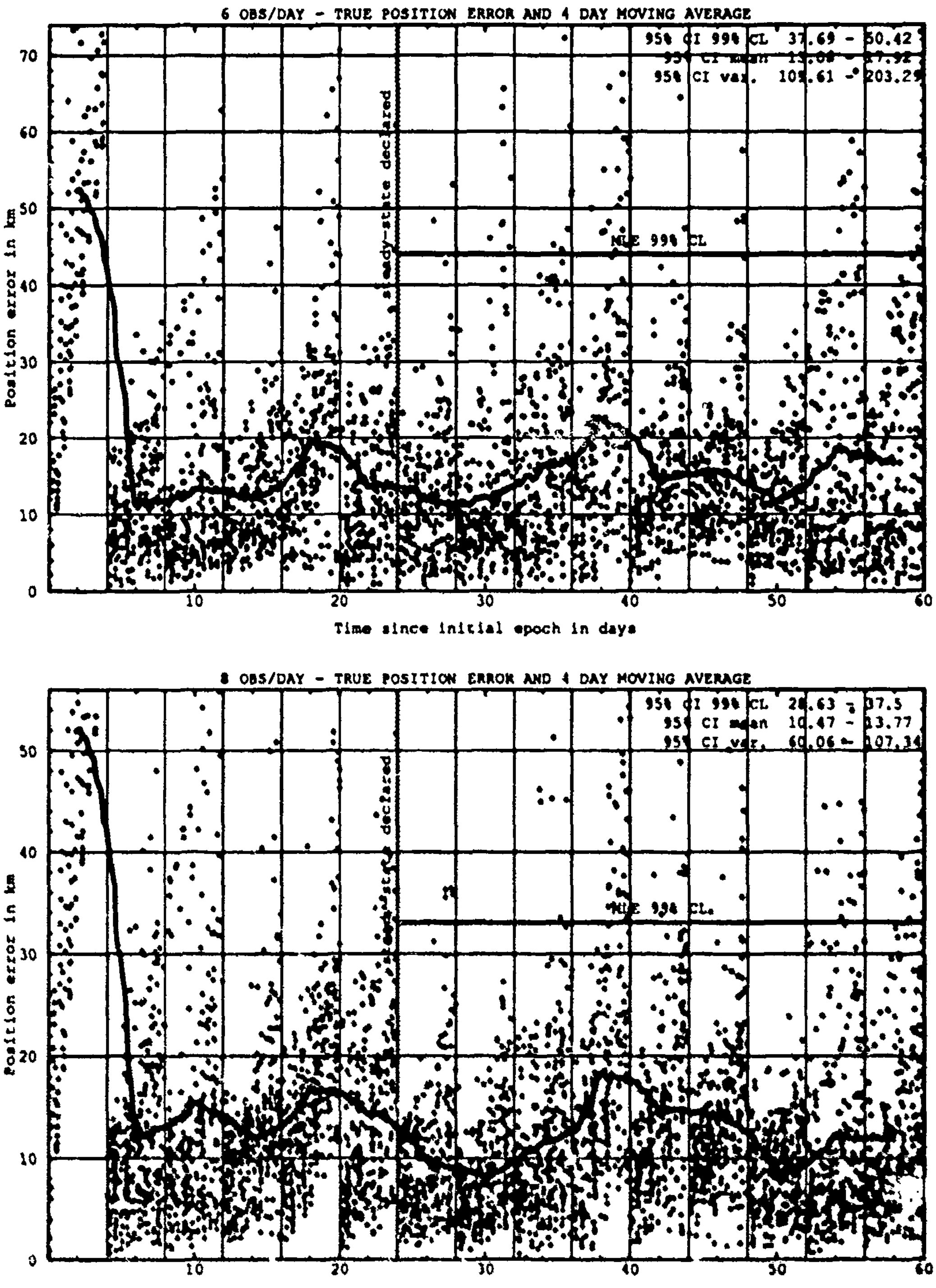


Figure II.22. Class: 1-3-2 (Catalog Number 14199), LUPI 4, OPD 6 and 8.

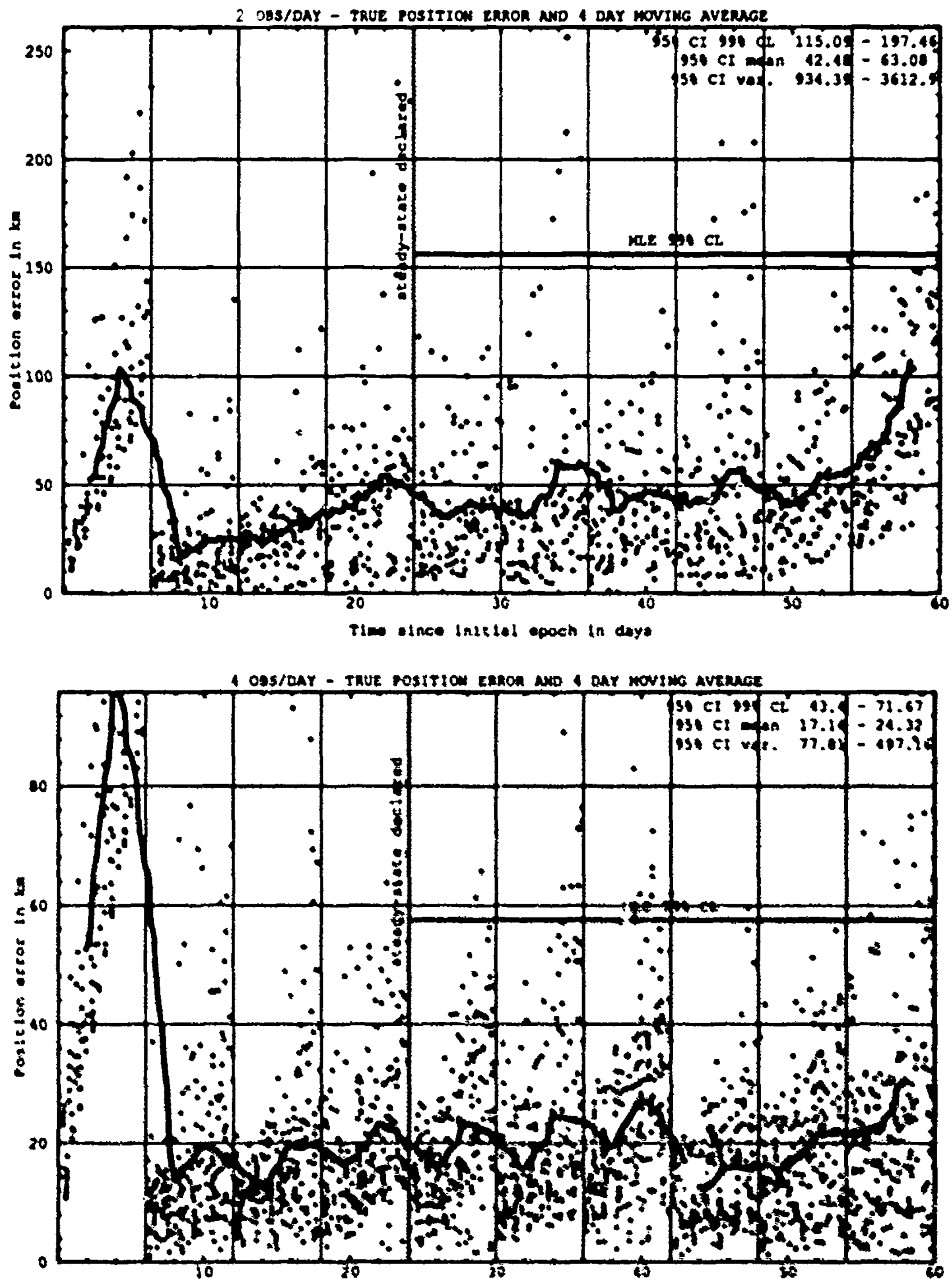


Figure H.23. Class: 1-3-2 (Catalog Number 14199), LUPI 6, OPD 2 and 4.

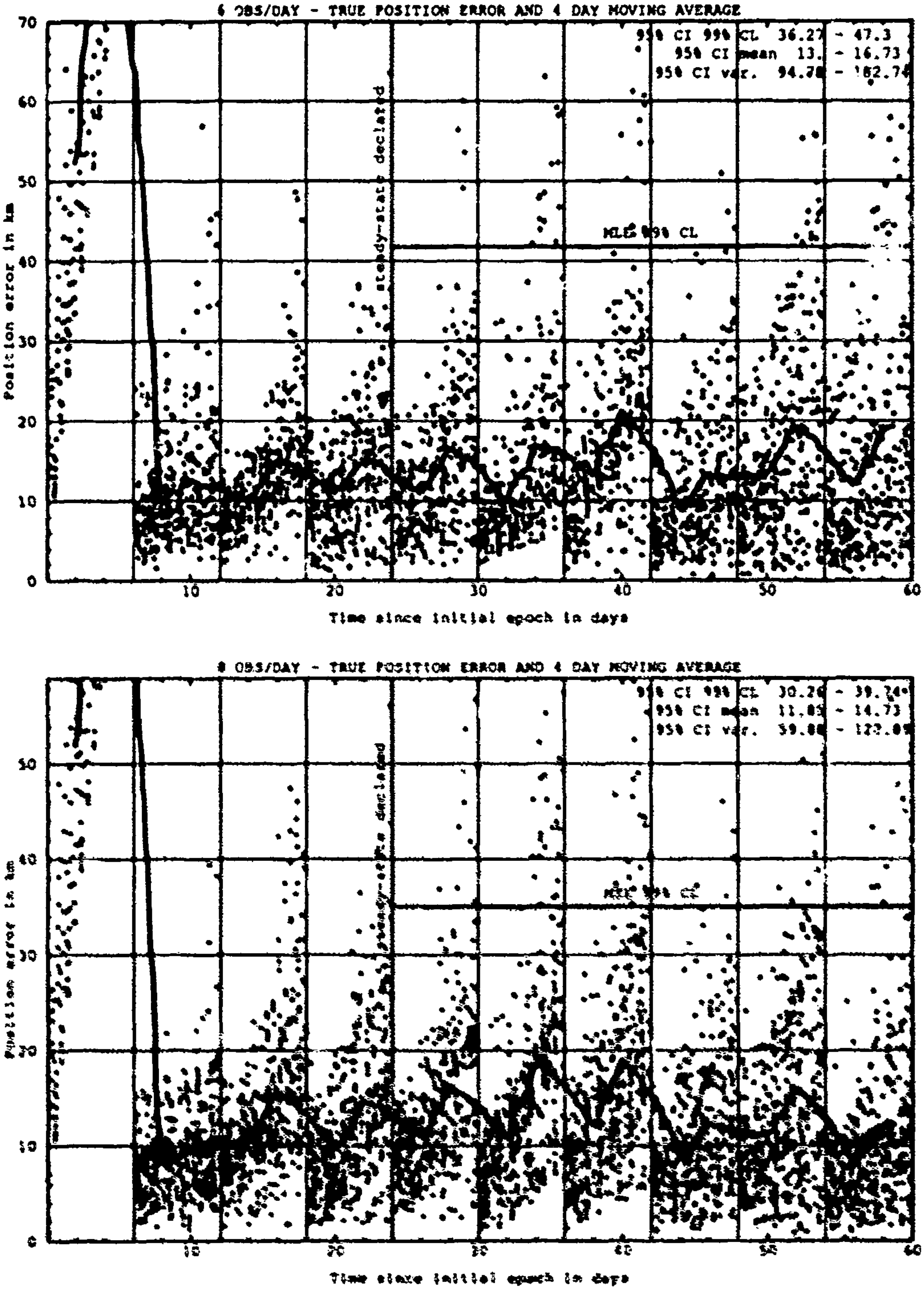
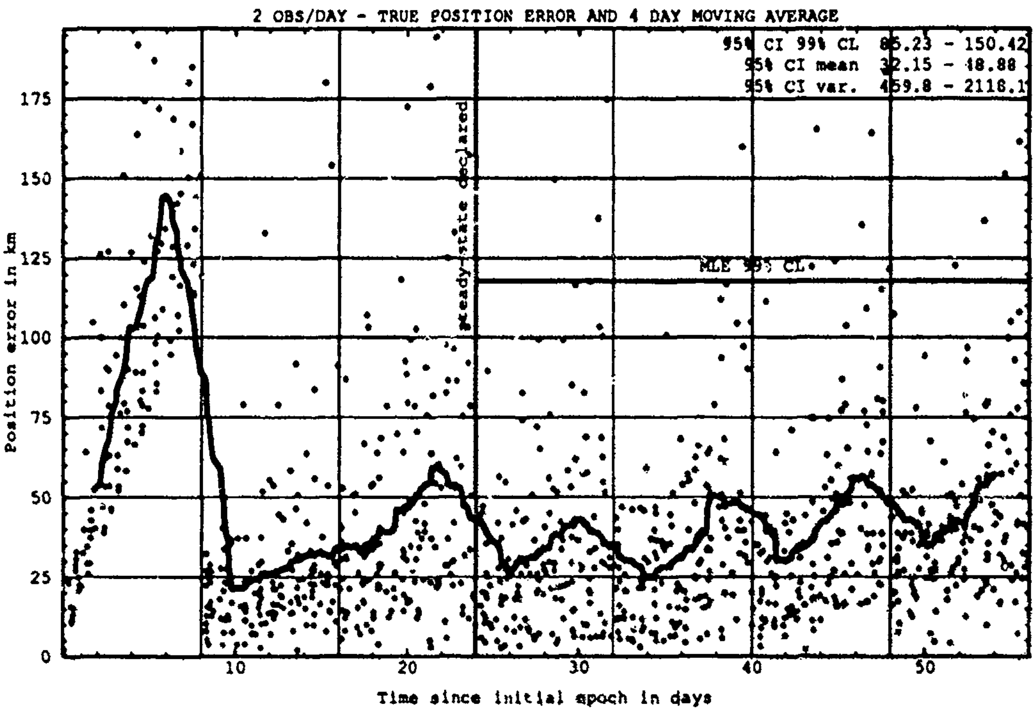


Figure H.24. Class: 1-3-2 (Catalog Number 14199), LUPI 6, OPD 6 and 8.



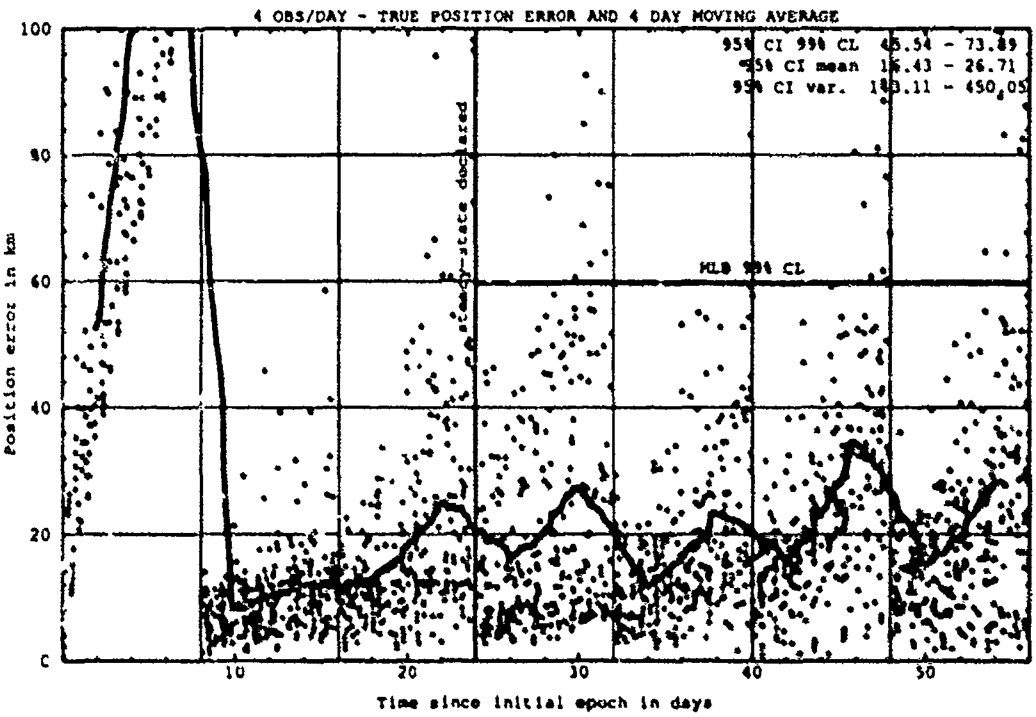


Figure II.25. Class: 1-3-2 (Catalog Number 14199), LUPI 8, OPD 2 and 4.

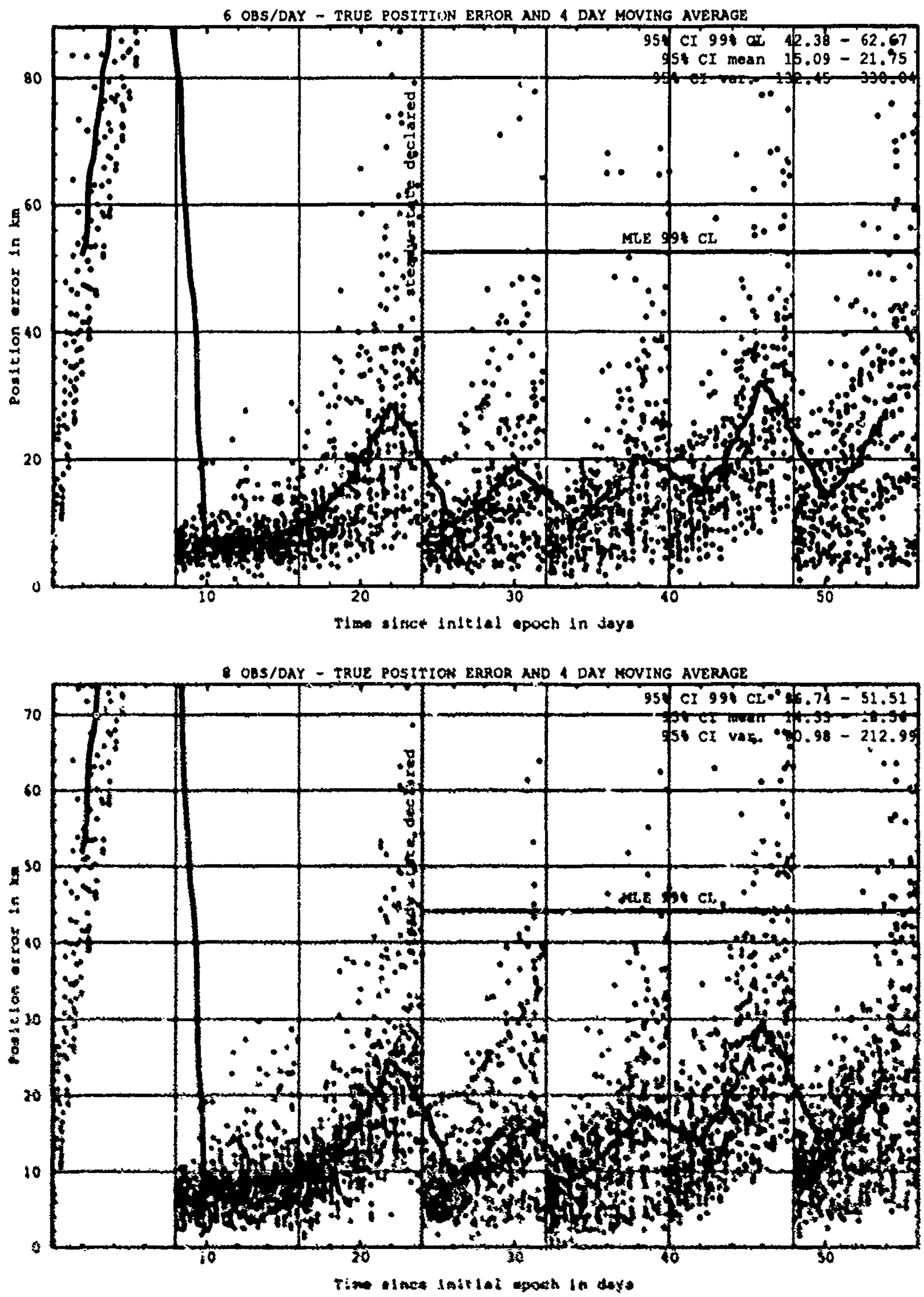


Figure H.26. Class: 1-3-2 (Catalog Number 14199), LUPI 8, OPD 6 and 8.

H.3.2 Differential Corrector Performance. The table below provides statistical information on the last-pass residuals of the differential corrector.

Table H.6. Selected Last-Pass 95% Confidence Interval Statistics.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
14199	8	8	6.34	8.15	7.52	19.16	12.92	17.98

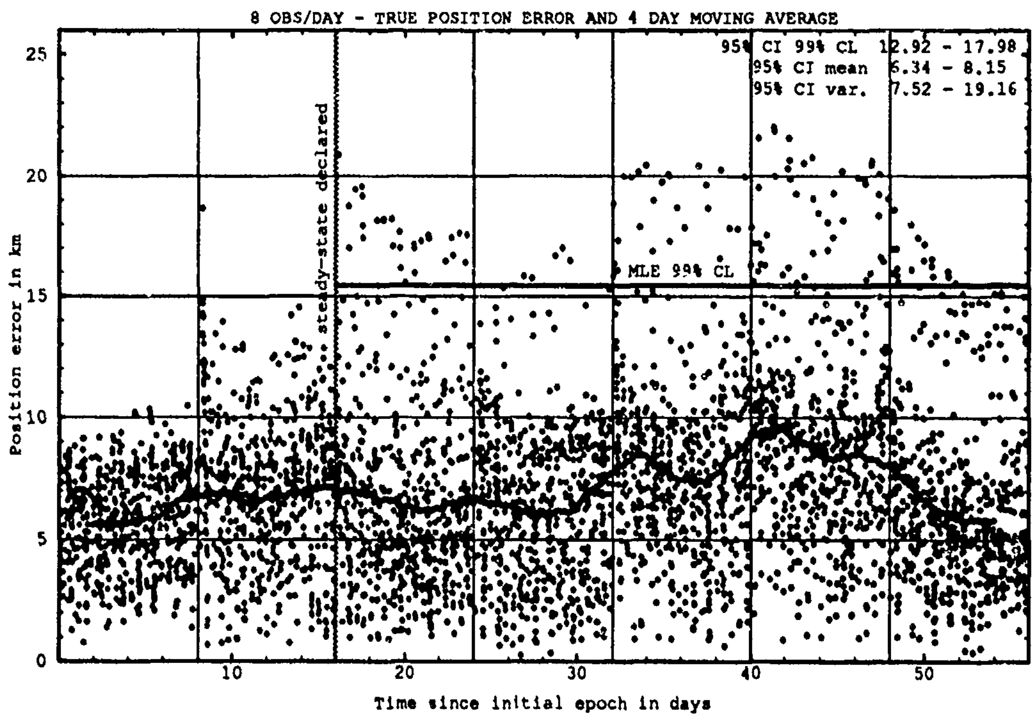


Figure H.27. Last-Pass — Class: 1-3-2 (Catalog Number 14199), LUPI 8, OPD 8.

H.4 CLASS: 2-1-3 (NORAD Catalog Number 10293)

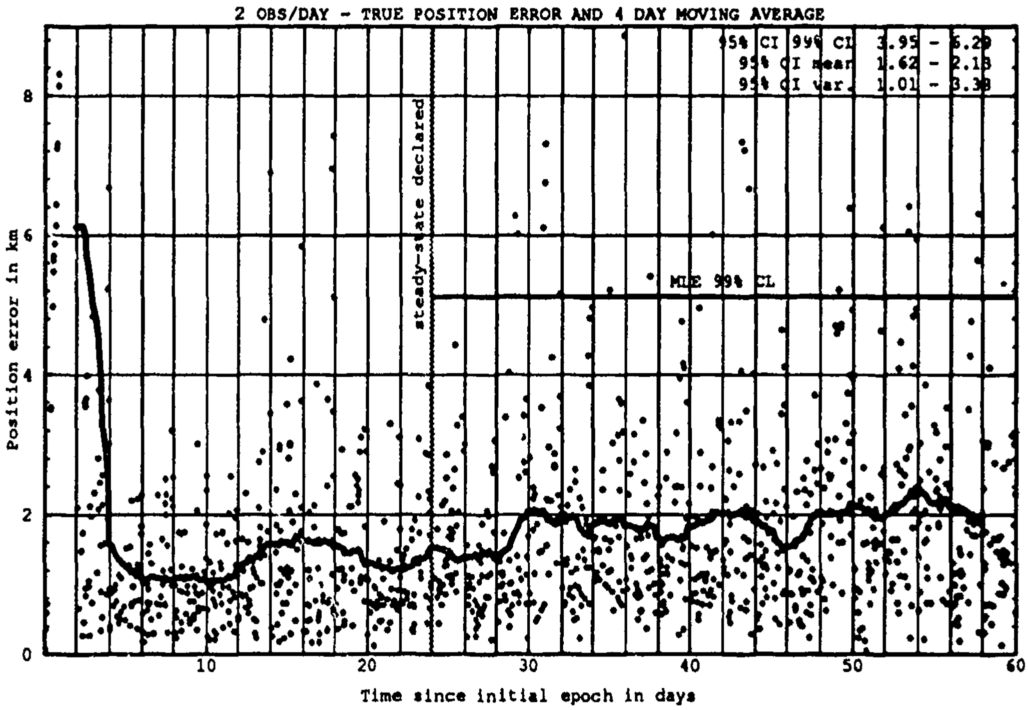
H.4.1 Prediction Performance. The tables below provide the statistical information on the ability of the estimates to predict for 1 LUPI after differential correction.

Table H.10. 95% Confidence Interval Analysis on Class: 2-1-3.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
10293	2	2	1.62	2.13	1.01	3.38	3.95	6.29
10293	2	4	1.28	1.46	0.66	1.02	3.17	3.79
10293	2	6	1.20	1.47	0.68	1.33	3.10	4.14
10293	2	8	1.16	1.36	0.71	1.47	3.09	4.15
10293	4	2	2.10	2.74	2.26	5.51	5.43	8.18
10293	4	4	2.01	2.73	1.80	4.73	5.25	7.62
10293	4	6	2.03	2.32	2.16	2.85	5.48	6.20
10293	4	8	1.85	2.16	1.62	2.92	4.85	6.08
10293	6	2	2.80	3.95	3.70	10.10	7.31	11.11
10293	6	4	2.35	2.96	2.18	4.52	5.92	7.73
10293	6	6	2.04	2.66	1.32	3.61	4.80	6.91
10293	6	8	1.86	2.42	1.01	2.15	4.17	5.79
10293	8	2	2.58	3.82	1.76	7.78	5.92	9.94
10293	8	4	2.08	2.81	1.24	3.57	4.79	7.04
10293	8	6	1.90	2.56	0.94	2.58	4.19	6.19
10293	8	8	1.80	2.33	0.55	2.74	3.80	5.94

Table H.11. ANOVA Analysis on Class: 2-1-3.

Source of Variation	Degrees of Freedom	Sum of Square	Mean Square	F	Table F
Replications	9	17.06	1.89	0.8971	1.88
Main Effects:					
LUPI	3	172.99	57.66	26.74	2.60
OPD	3	148.61	49.54	22.98	2.60
Interaction	9	36.53	4.06	1.88	1.88
Error	135	291.07	2.16		
Total	159	667.27			



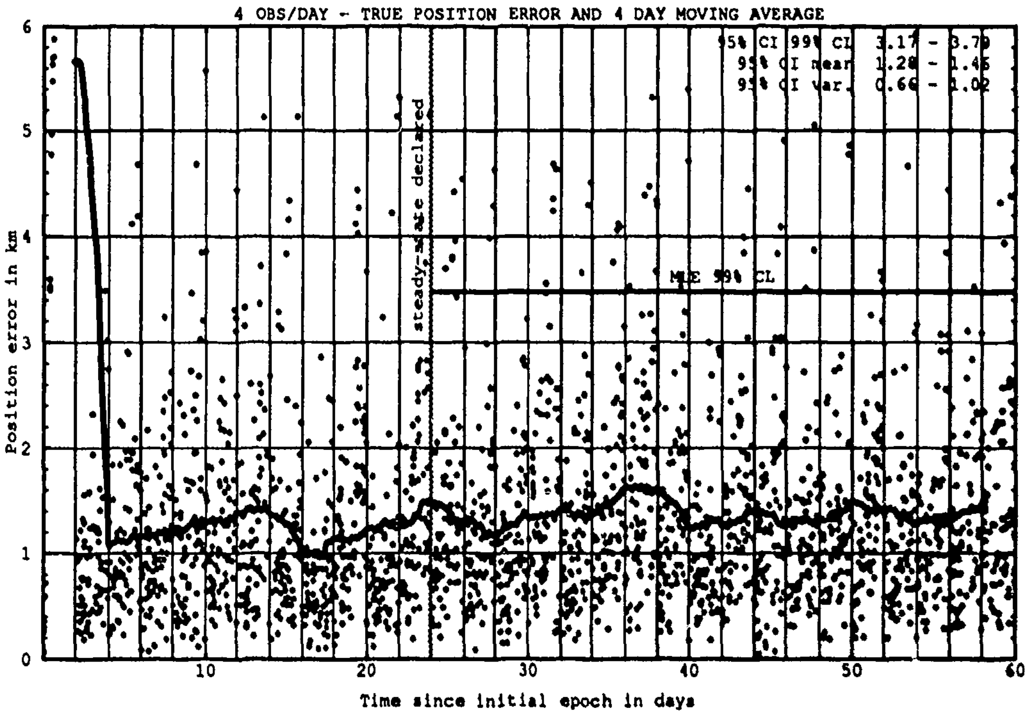
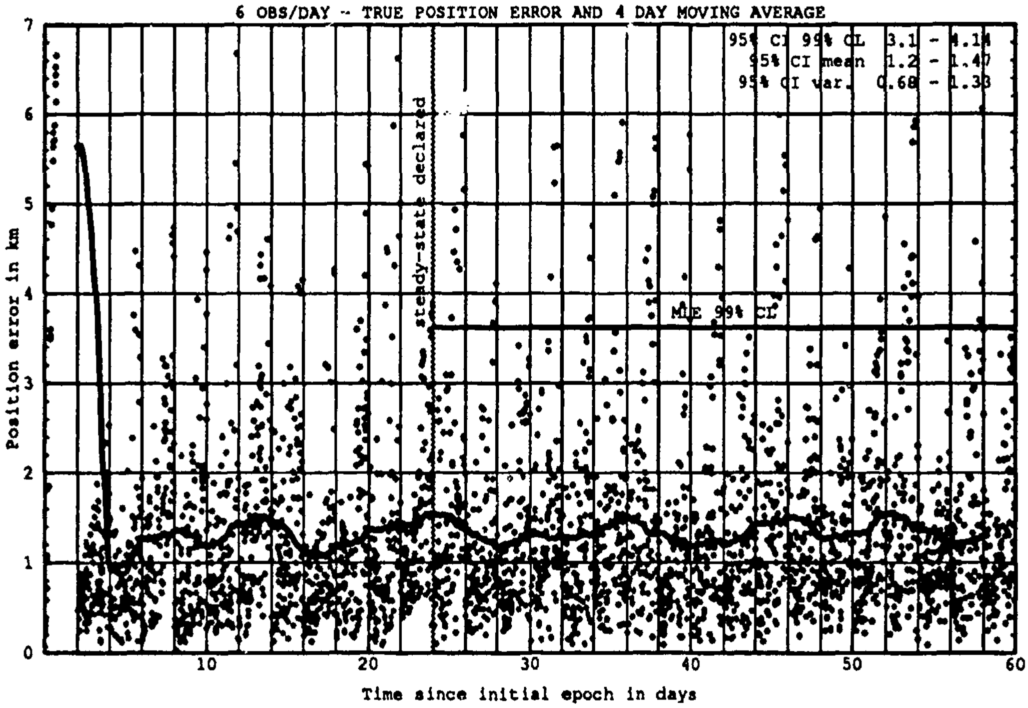


Figure H.28. Class: 2-1-3 (Catalog Number 10293), LUPI 2, OPD 2 and 4.



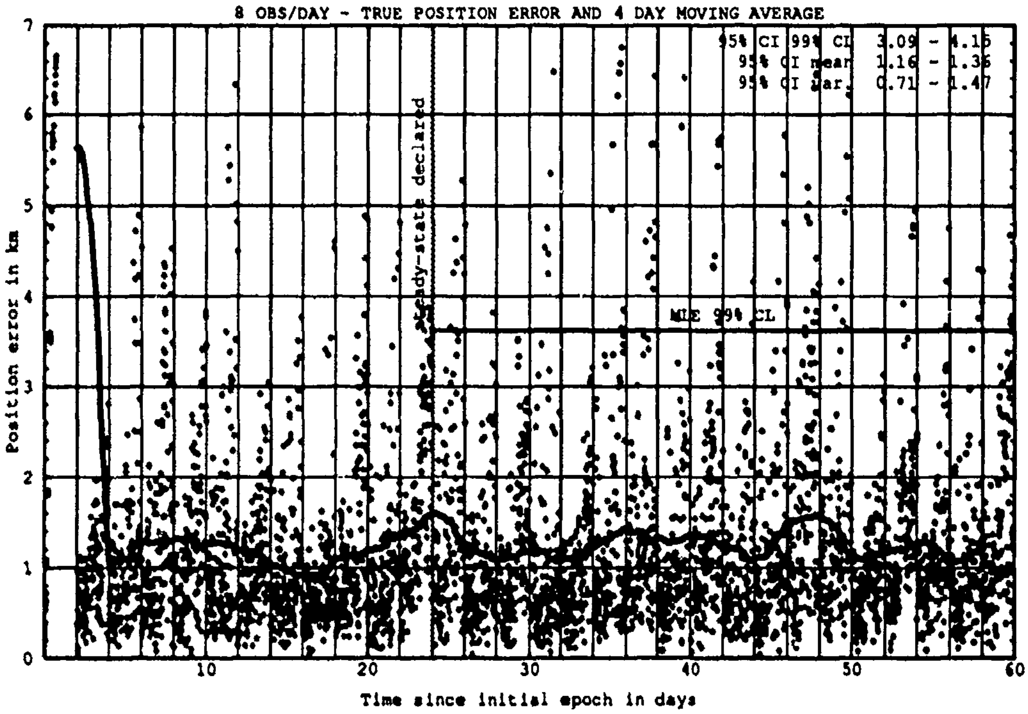
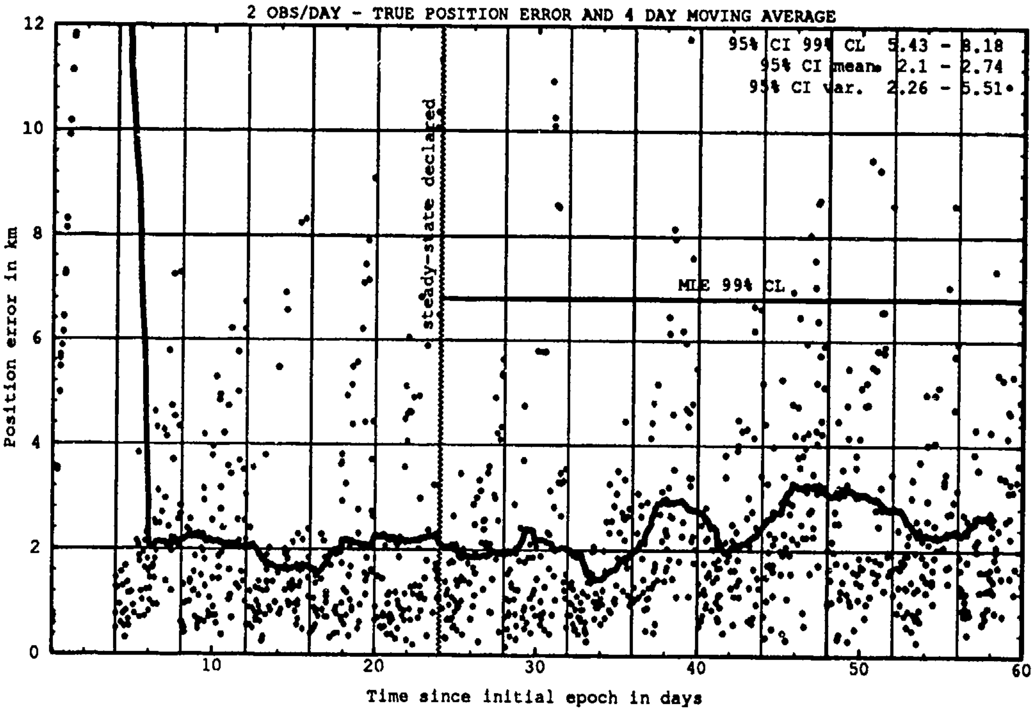


Figure H.29. Class: 2-1-3 (Catalog Number 10293), LUPI 2, OPD 6 and 8.



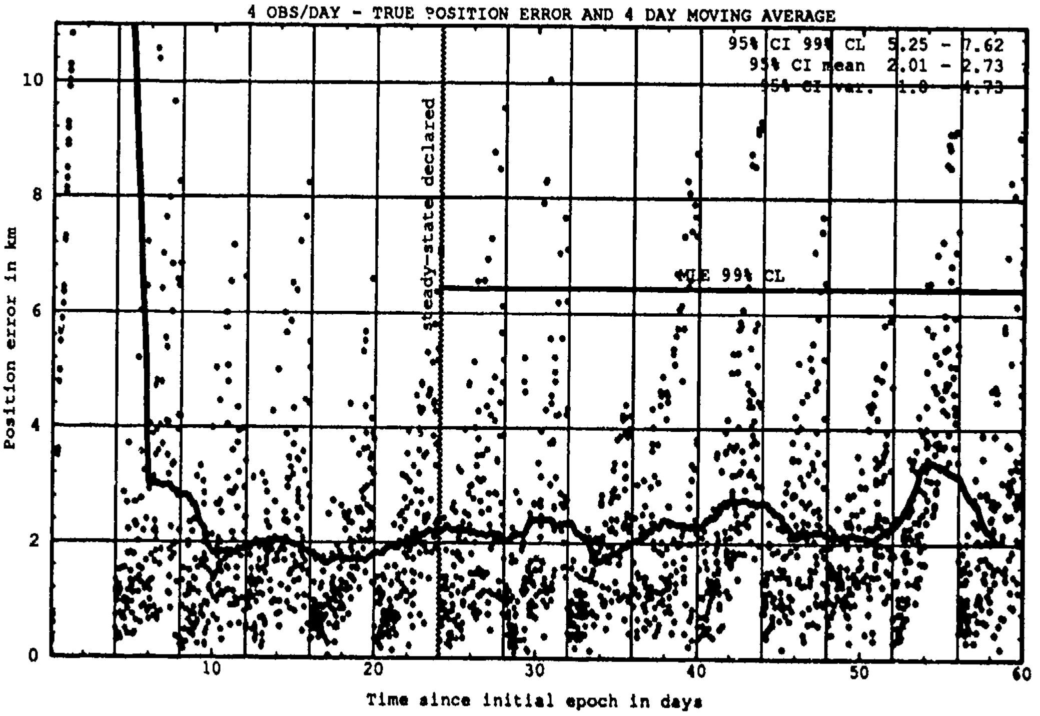
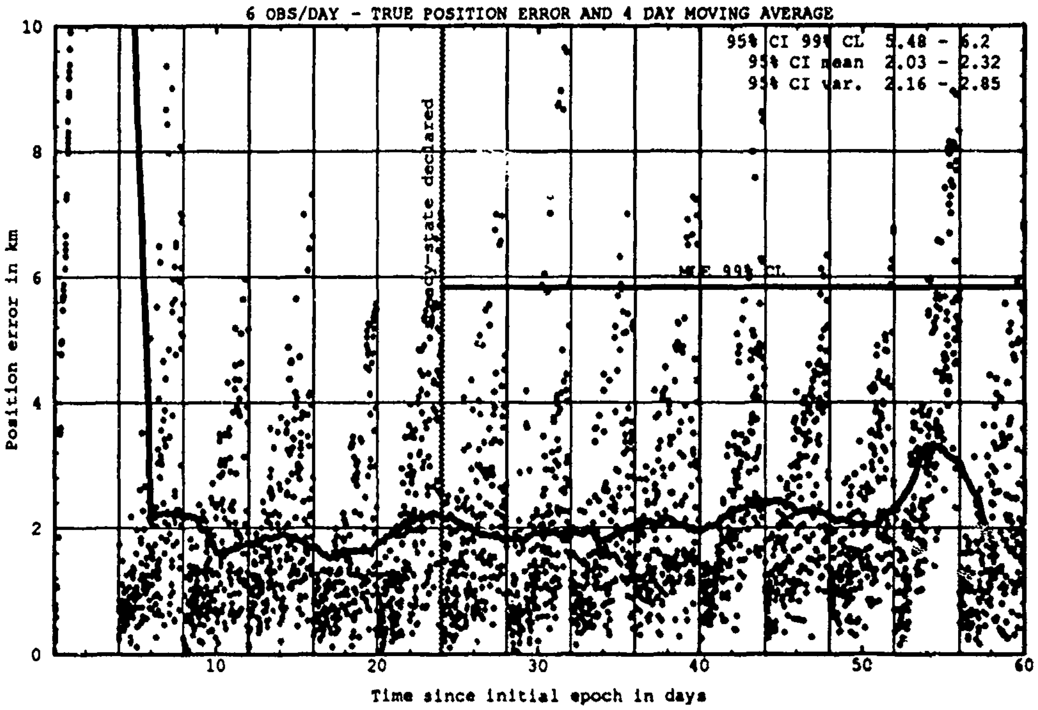


Figure H.30. Class: 2-1-3 (Catalog Number 10293), LUPI 4, OPD 2 and 4.



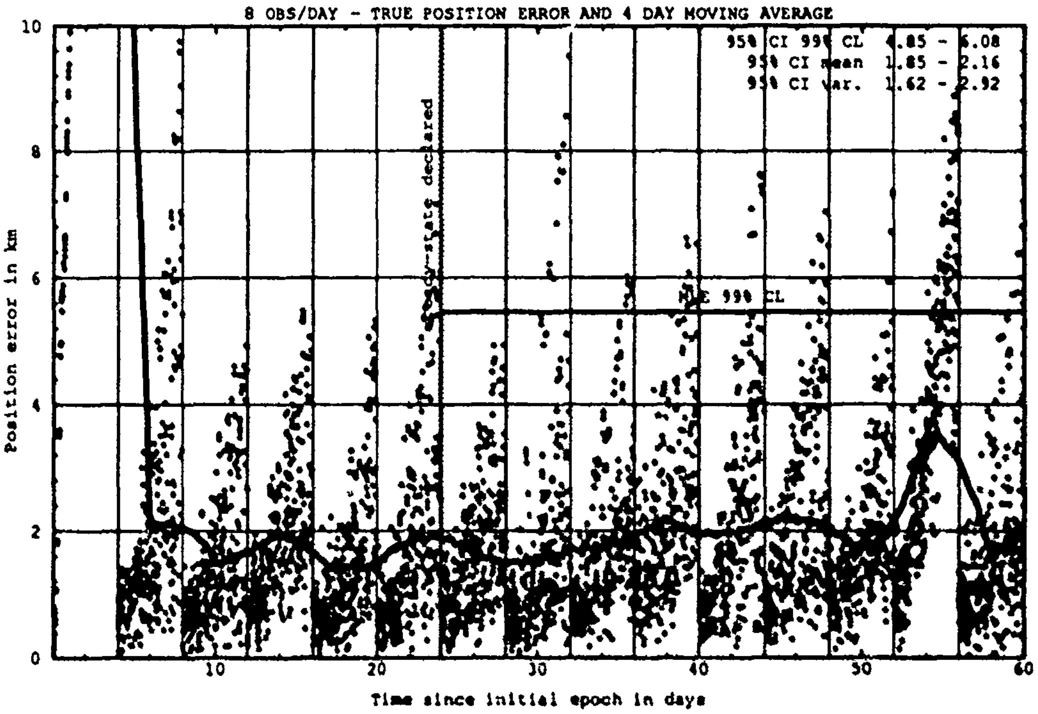


Figure H.31. Class: 2-1-3 (Catalog Number 10293), LUPI 4, OPD 6 and 8.

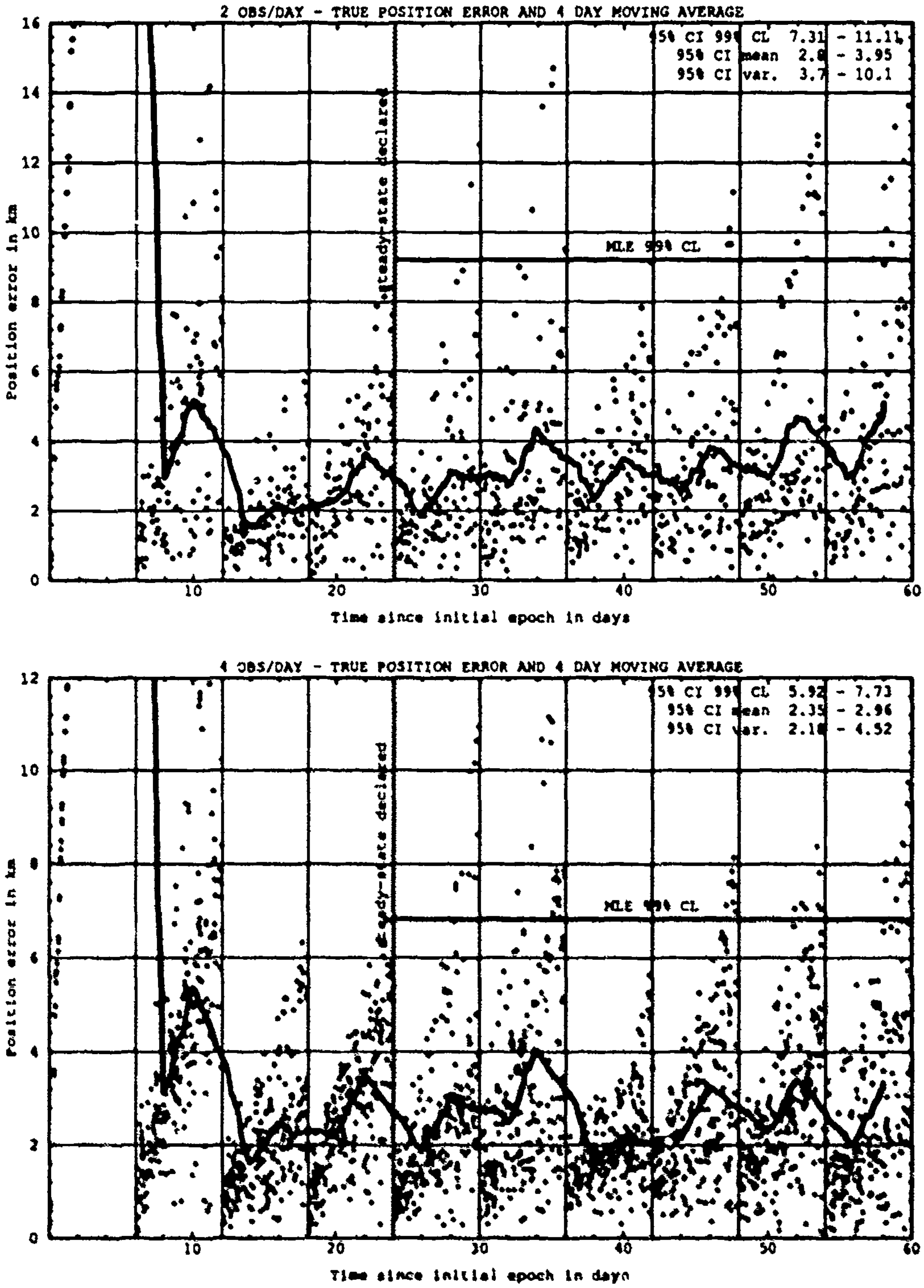


Figure H.32. Class: 2-1-3 (Catalog Number 10293), LUPI 6, OPD 2 and 4.

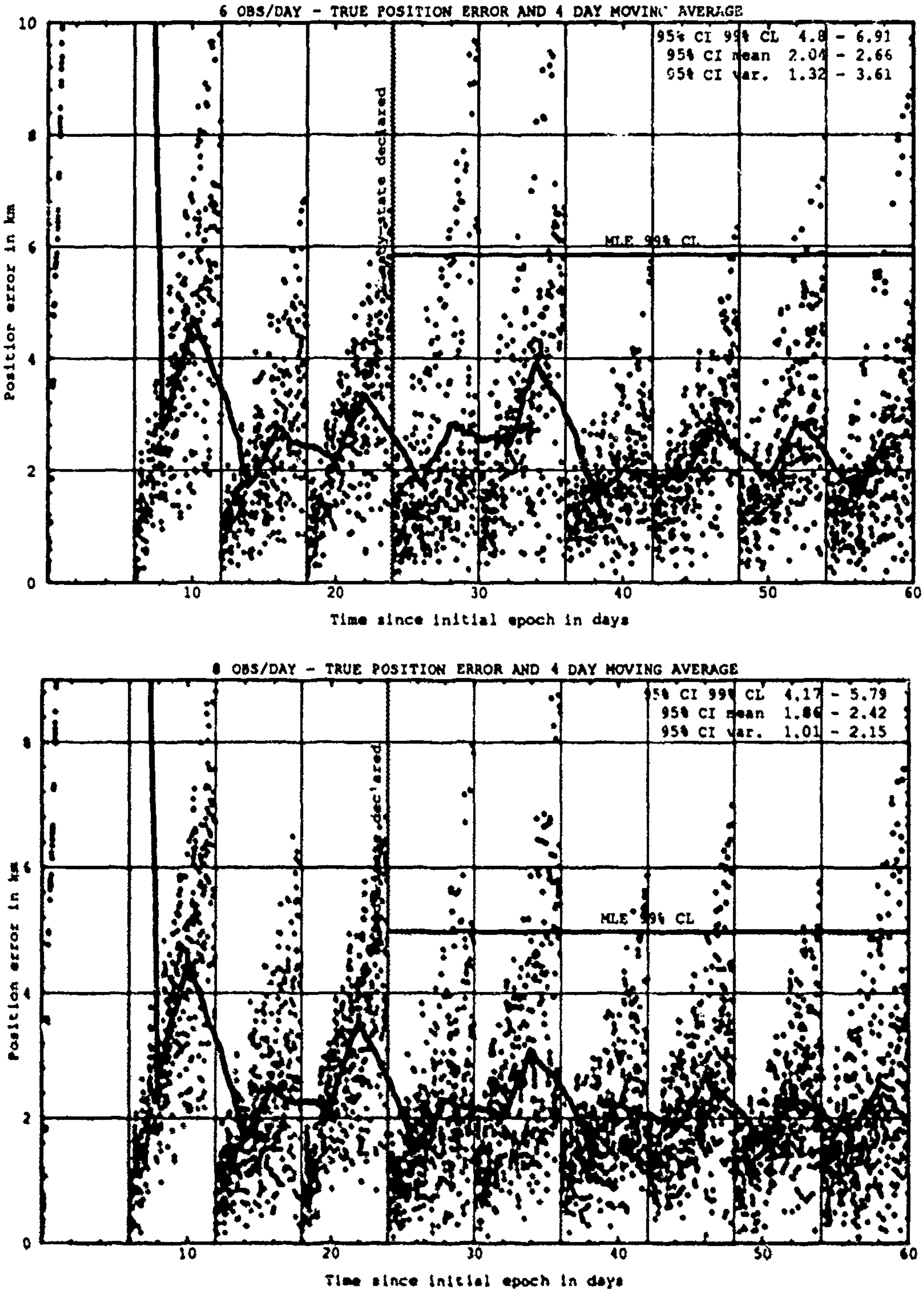
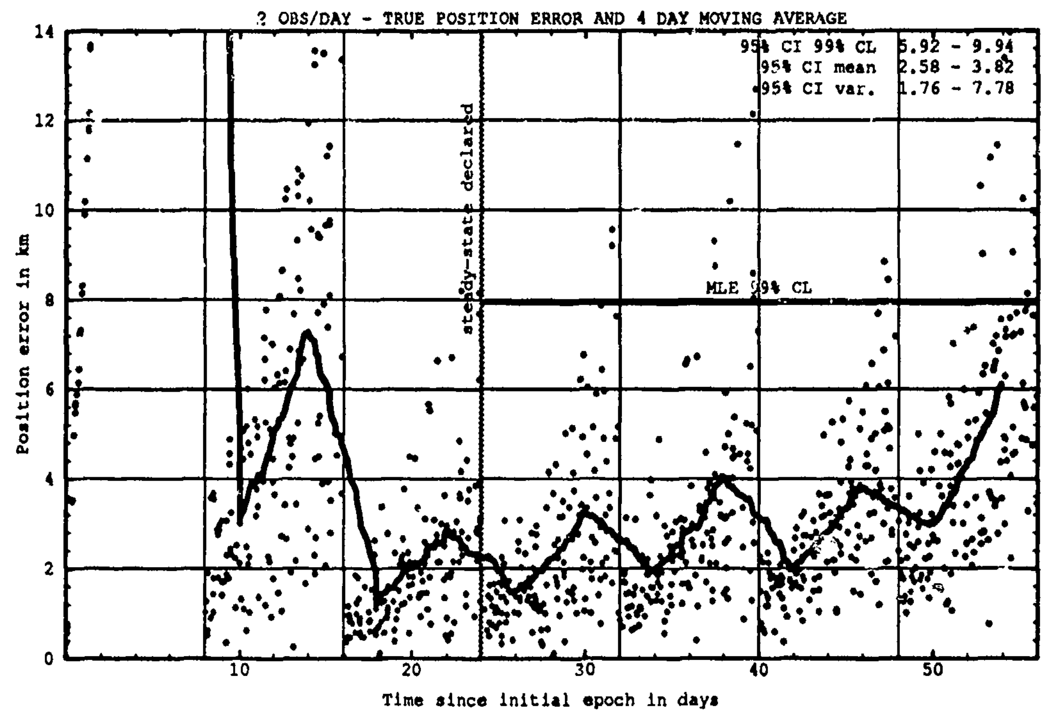


Figure H.33. Class: 2-1-3 (Catalog Number 10293), LUPI 6, OPD 6 and 8.



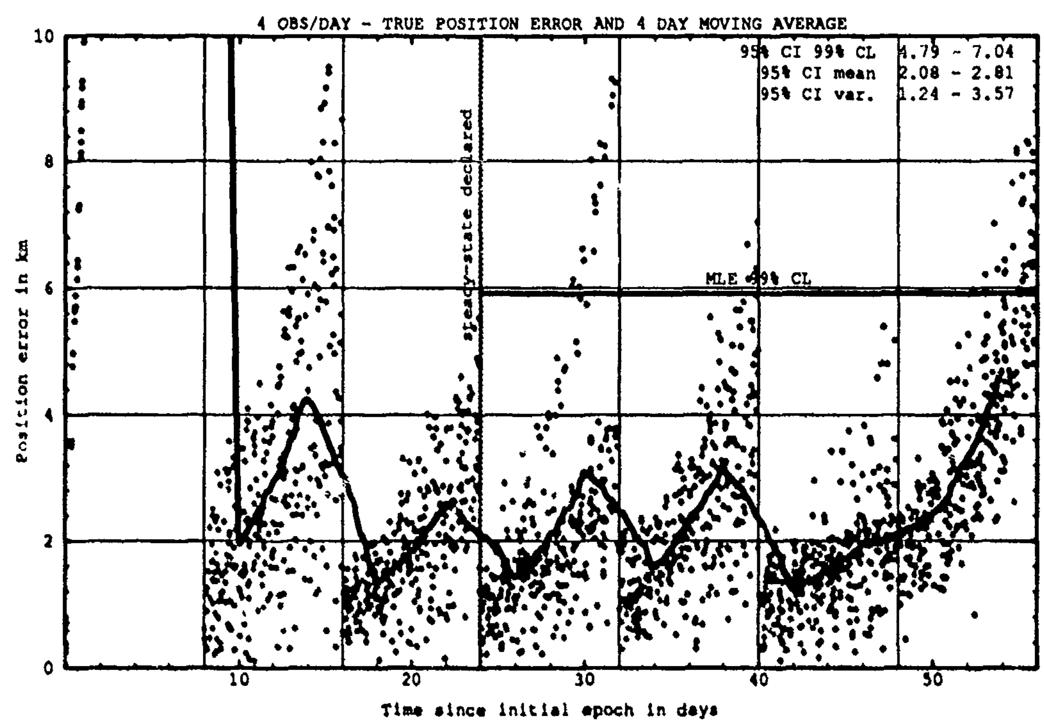


Figure H.34. Class: 2-1-3 (Catalog Number 10293), LUPI 8, OPD 2 and 4.

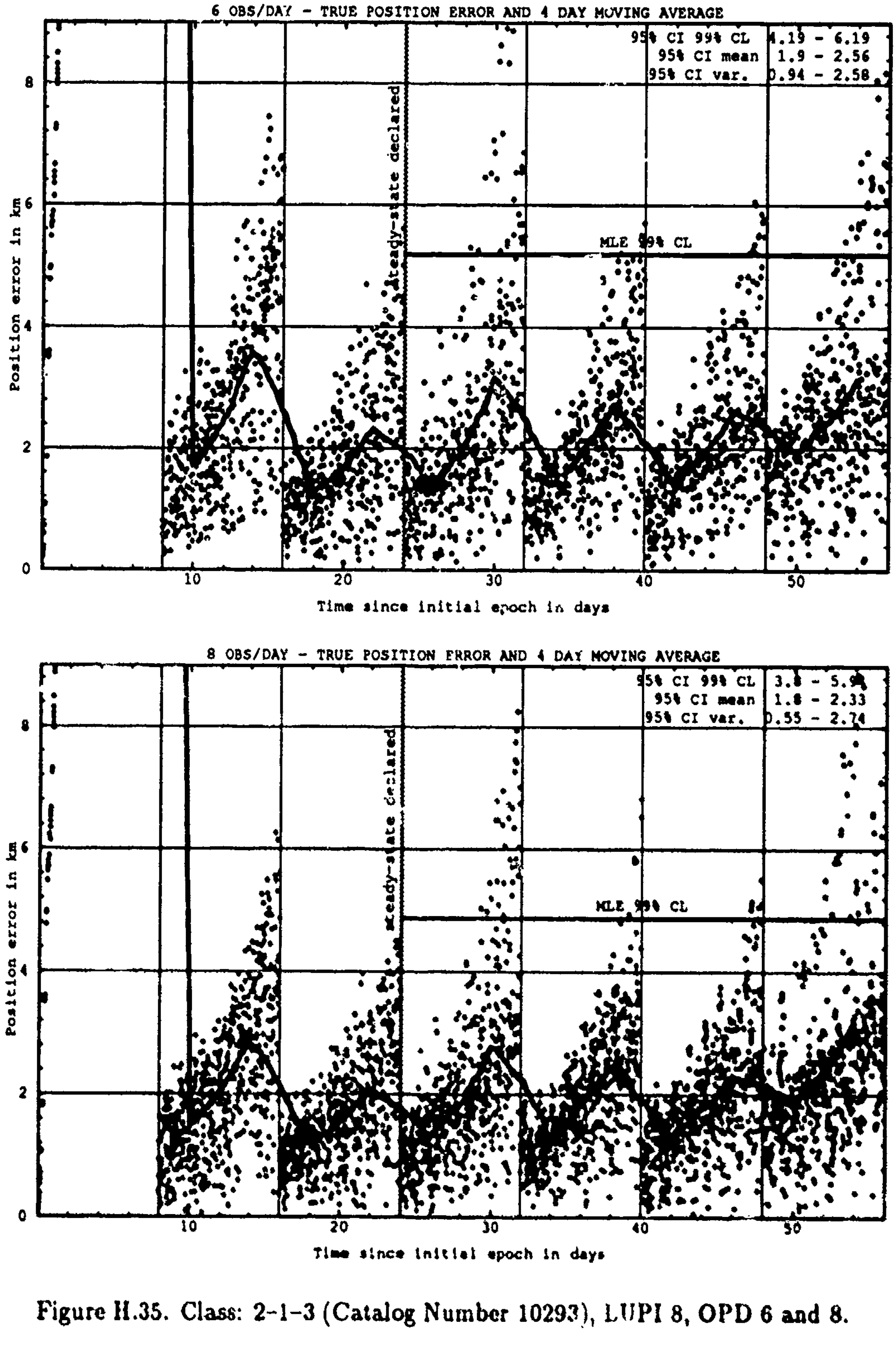


Figure H.35. Class: 2-1-3 (Catalog Number 10293), LUPI 8, OPD 6 and 8.

H.4.2 Differential Corrector Performance. **The table below provides statistical information on the last-pass residuals of the differential corrector.**

Table H.8. Selected Last-Pass 95% Confidence Interval Statistics.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
10293	8	8	0.64	0.68	0.10	0.13	1.37	1.51

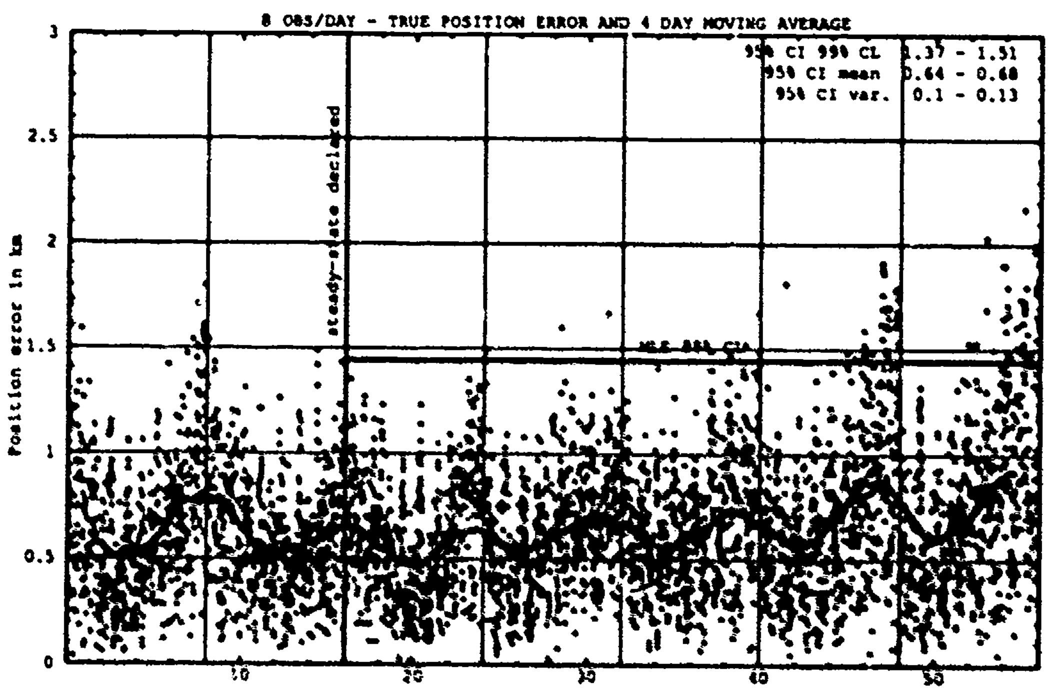


Figure H.36. Last-Pass — Class: 2-1-3 (Catalog Number 10293), LUPI 8, OPD 8.

H.5 CLASS: 2-1-4 (NORAD Catalog Number 10393)

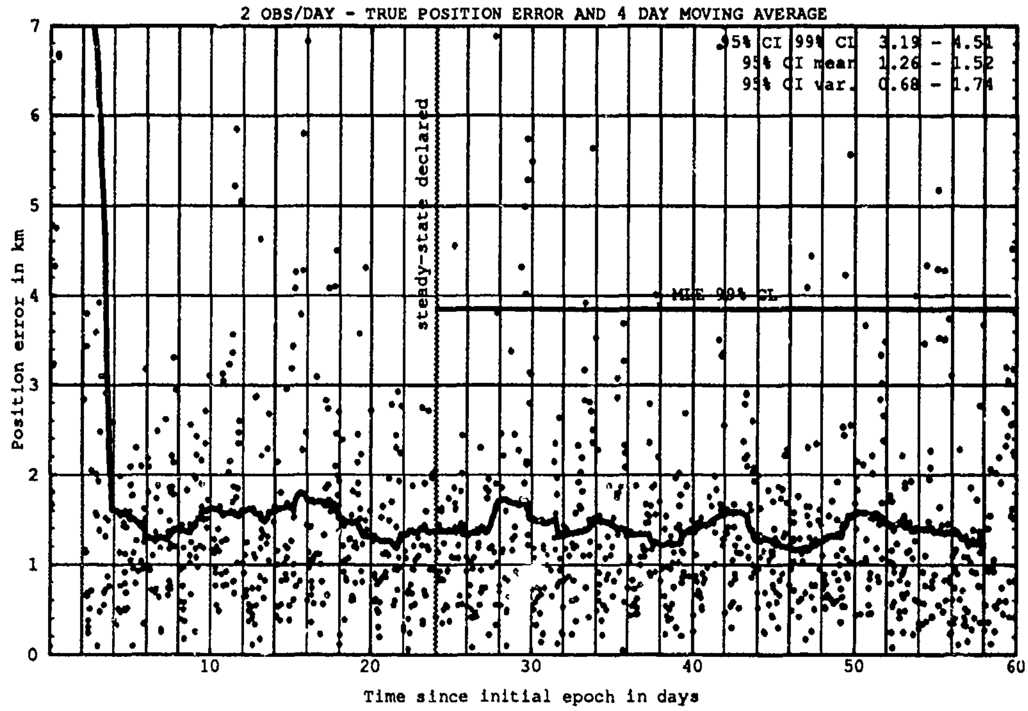
H.5.1 Prediction Performance. The tables below provide the statistical information on the ability of the estimates to predict for 1 LUPI after differential correction.

Table H.13. 95% Confidence Interval Analysis on Class: 2-1-4.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
10393	2	2	1.26	1.52	0.68	1.74	3.19	4.51
10393	2	4	1.03	1.31	0.46	0.96	2.60	3.56
10393	2	6	0.96	1.23	0.43	0.77	2.45	3.27
10393	2	8	1.01	1.19	0.47	0.76	2.63	3.18
10393	4	2	1.70	1.99	1.19	1.91	4.30	5.13
10393	4	4	1.27	1.66	0.76	1.21	3.30	4.19
10393	4	6	1.22	1.50	0.71	1.17	3.19	3.99
10393	4	8	1.18	1.36	0.61	0.94	3.01	3.59
10393	6	2	1.93	2.93	1.01	5.14	4.57	7.89
10393	6	4	1.51	2.14	0.65	2.65	3.55	5.73
10393	6	6	1.46	1.93	0.71	1.80	3.43	4.98
10393	6	8	1.49	1.87	0.74	1.42	3.50	4.60
10393	8	2	2.12	3.29	1.11	7.19	4.86	9.15
10393	8	4	1.96	2.44	1.13	2.86	4.48	6.26
10393	8	6	1.74	2.09	0.83	1.89	3.92	5.18
10393	8	8	1.69	1.98	0.66	1.58	3.69	4.78

Table H.14. ANOVA Analysis on Class: 2-1-4.

Source of Variation	Degrees of Freedom	Sum of Square	Mean Square	F	Table F
Replications	9	44.77	4.97	4.41	1.88
Main Effects:					
LUPI	3	107.71	35.90	31.83	2.60
OPD	3	81.38	27.13	24.05	2.60
Interaction	9	12.17	1.35	1.19	1.88
Error	135	152.29	1.13		
Total	159	398.32			



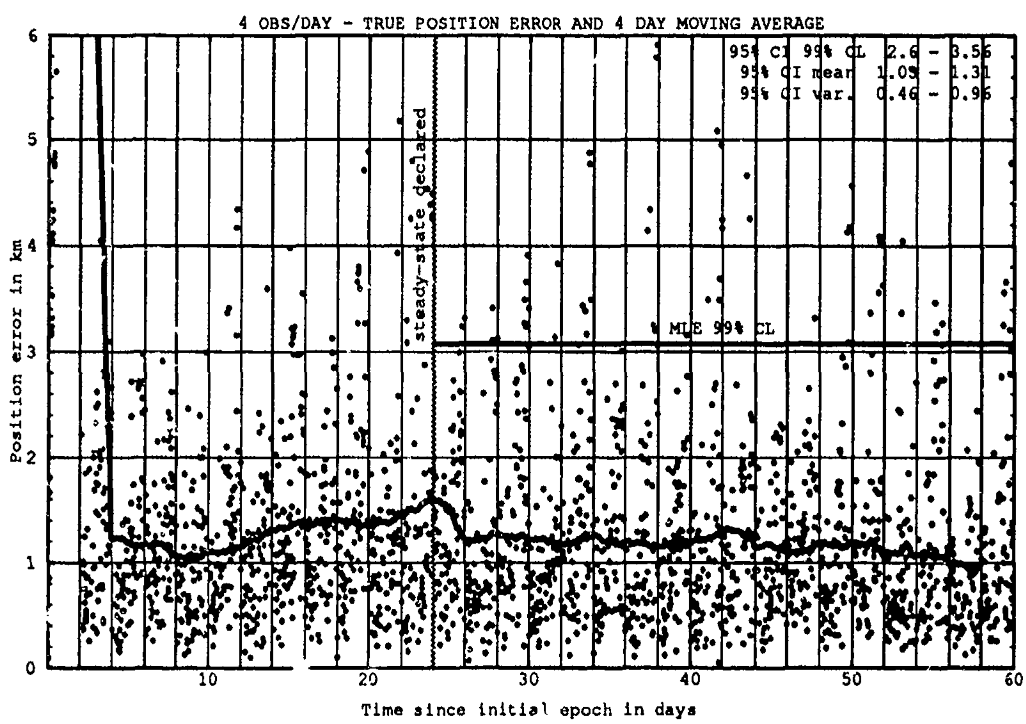


Figure H.37. Class: 2-1-4 (Catalog Number 10393), LUPI 2, OPD 2 and 4.

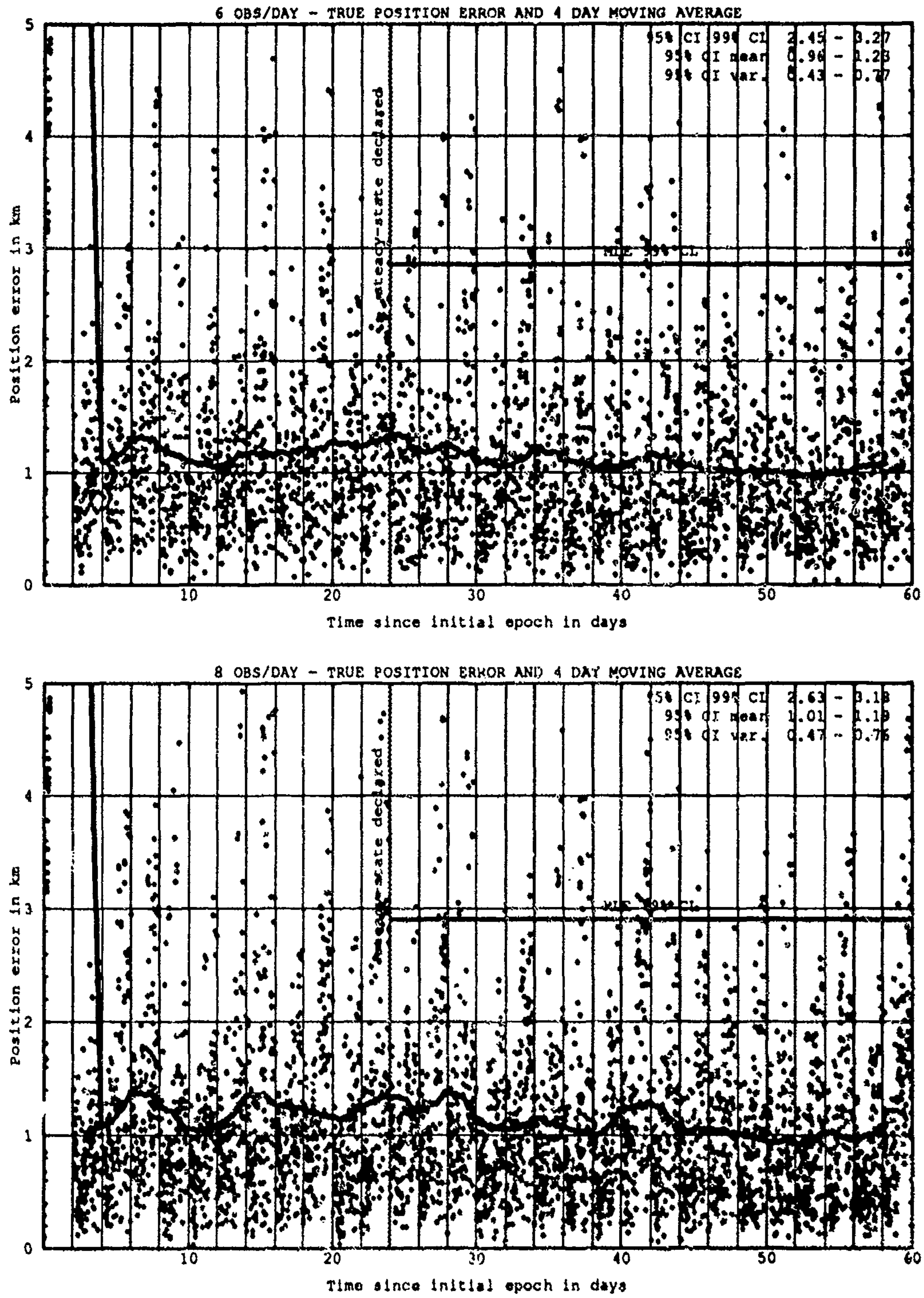
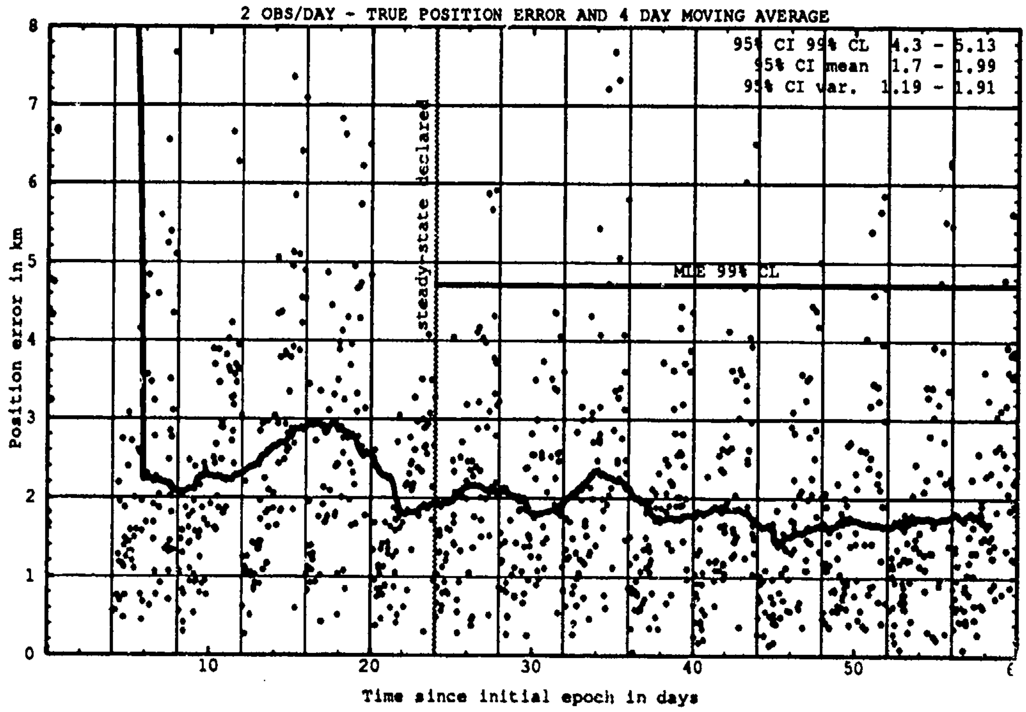


Figure H.38. Class: 2-1-4 (Catalog Number 10393), LUPI 2, OPD 6 and 8.



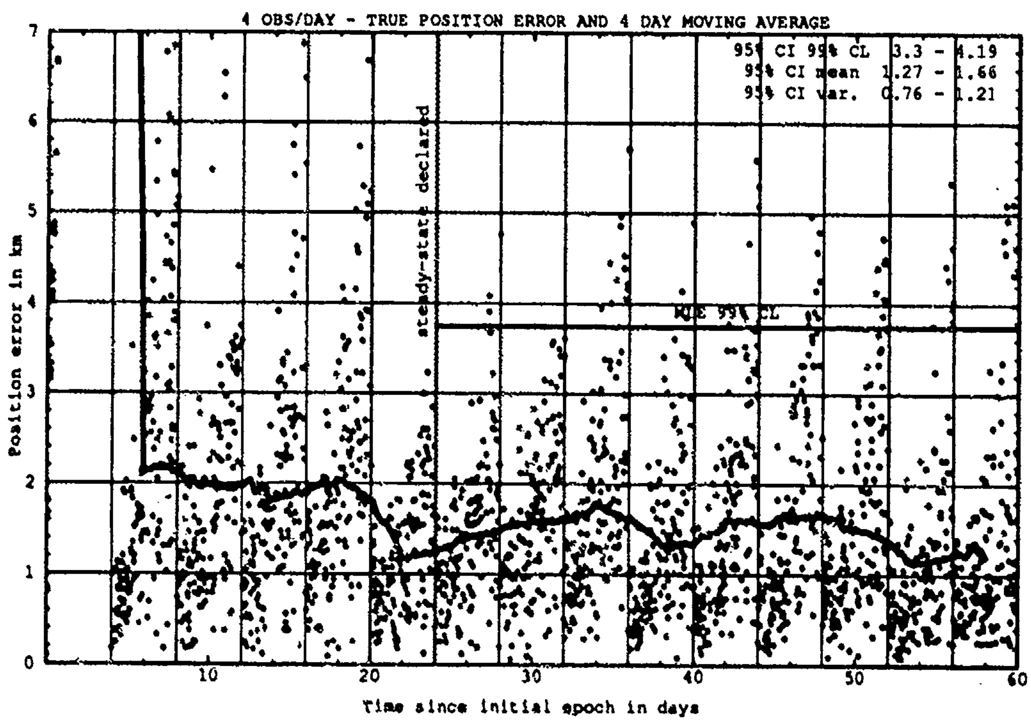
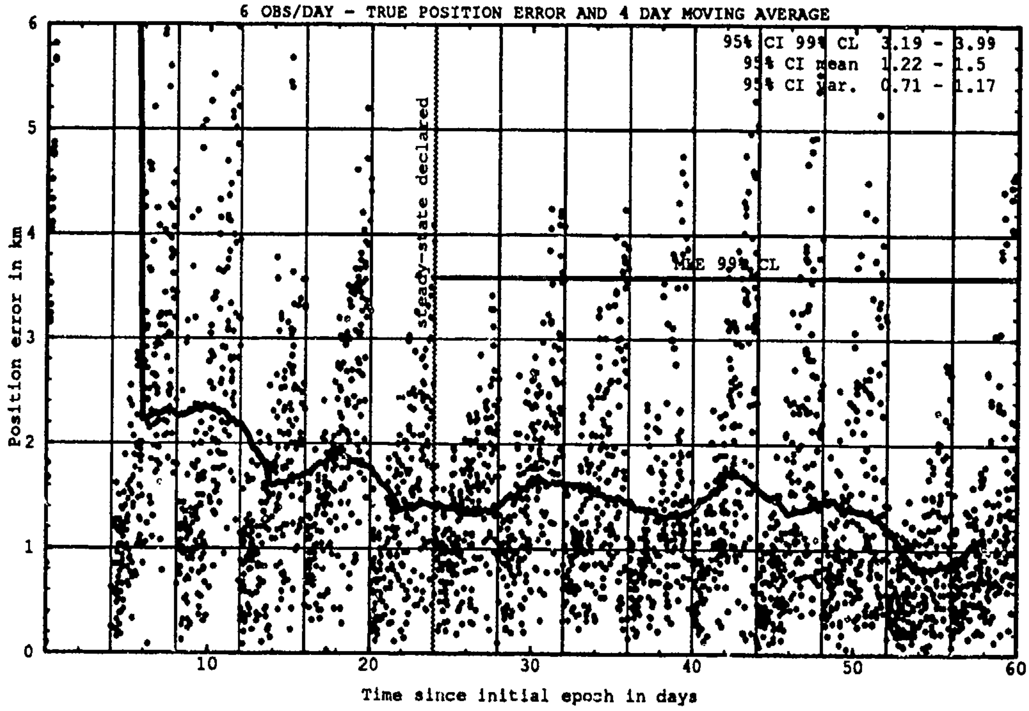


Figure H.39. Class: 2-1-4 (Catalog Number 10393), LUPI 4, OPD 2 and 4.



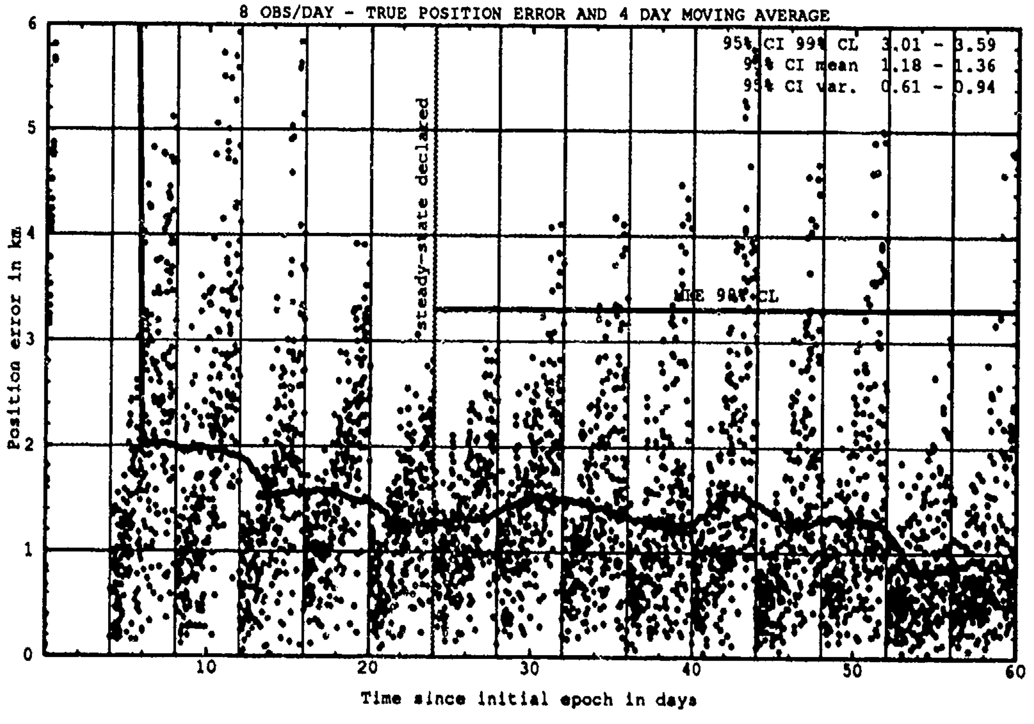
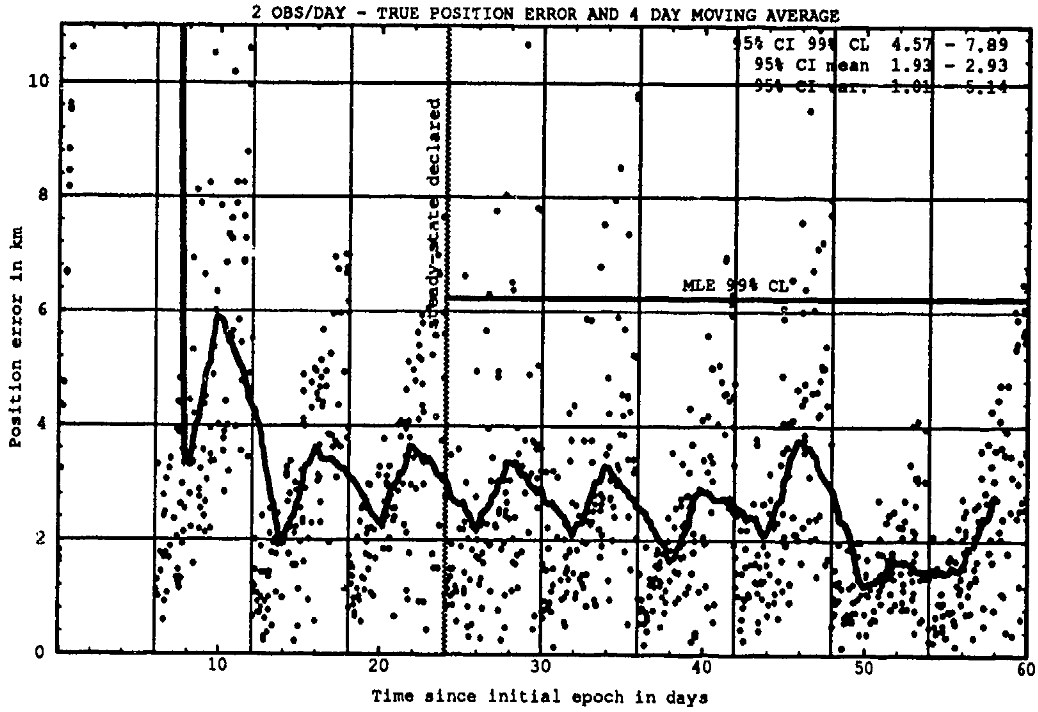


Figure H.40. Class: 2-1-4 (Catalog Number 10393), LUPI 4, OPD 6 and 8.



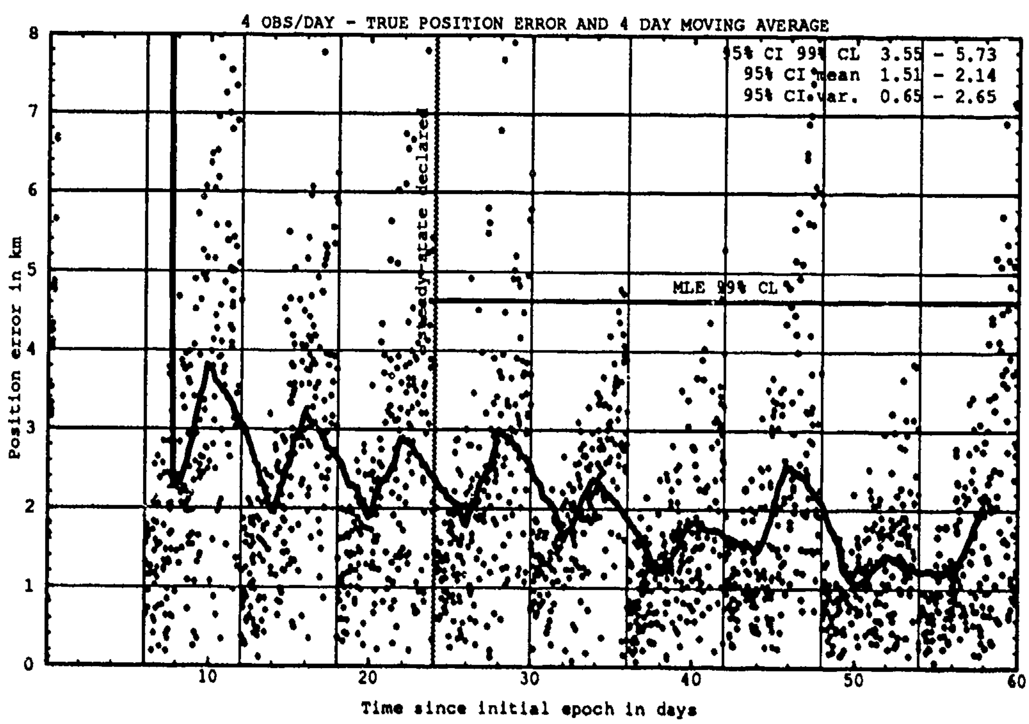


Figure H.41. Class: 2-1-4 (Catalog Number 10393), LUPI 6, OPD 2 and 4.

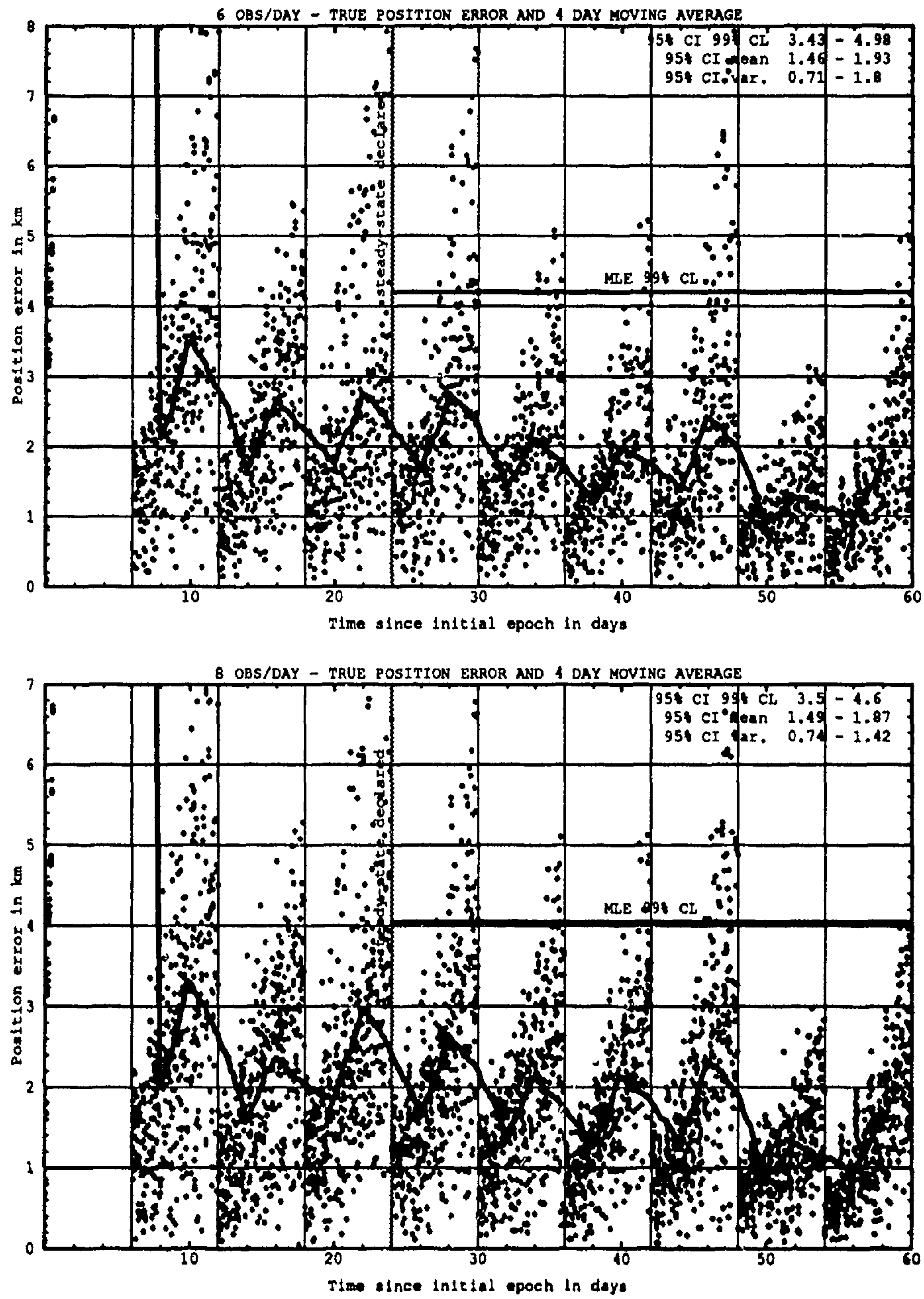
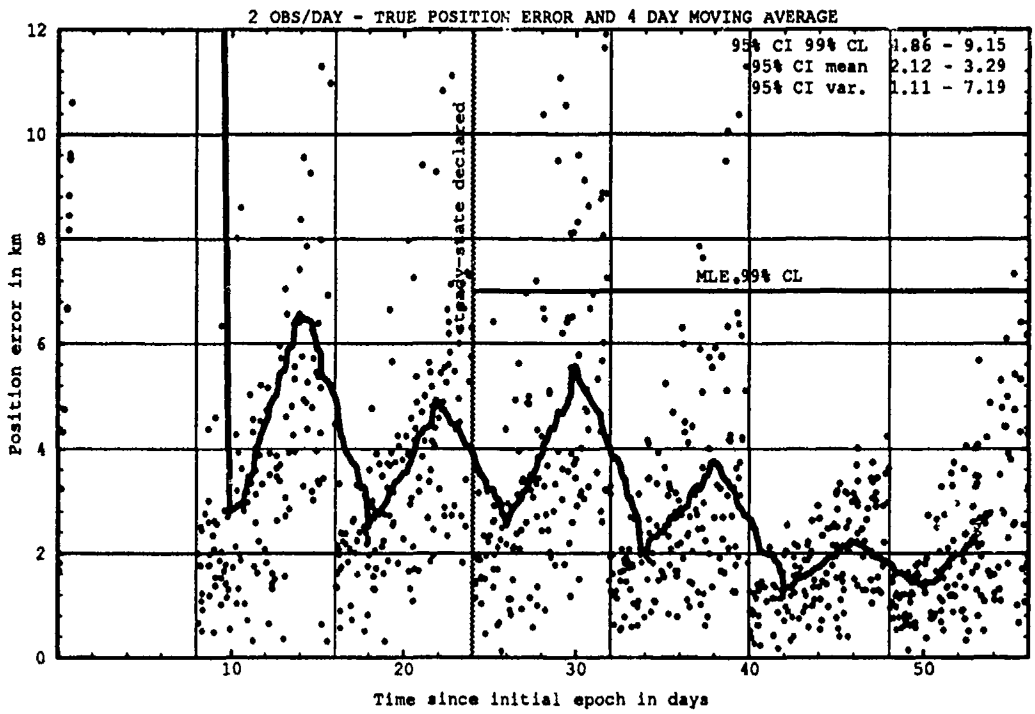


Figure H.42. Class: 2-1-4 (Catalog Number 10393), LUPI 6, OPD 6 and 8.



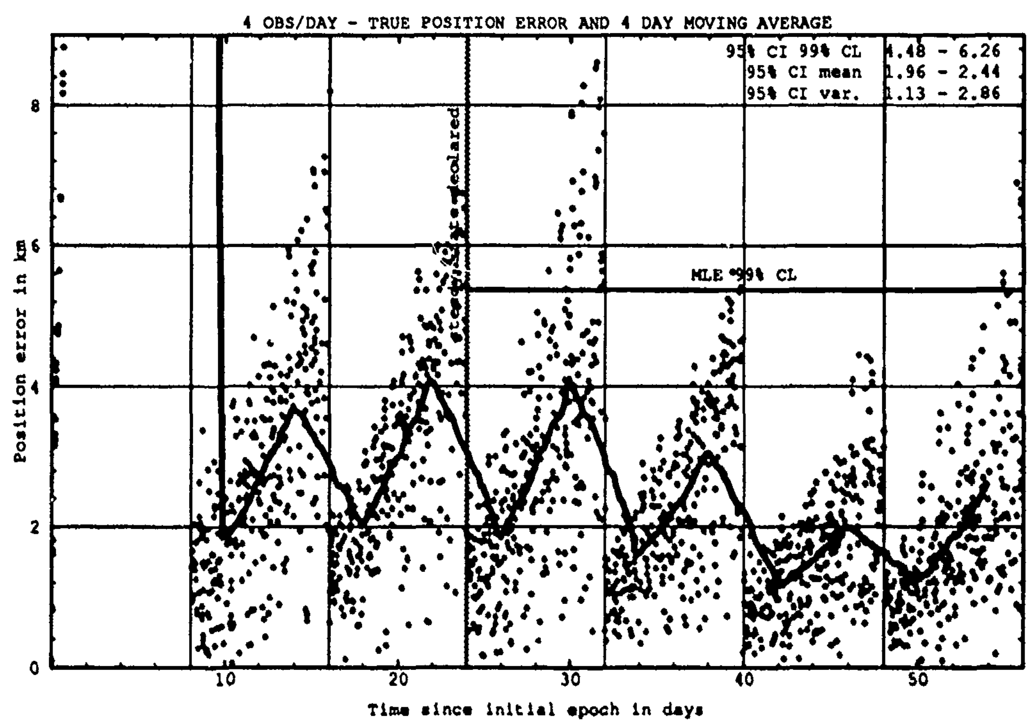


Figure H.43. Class: 2-1-4 (Catalog Number 10393), LUPI 8, OPD 2 and 4.

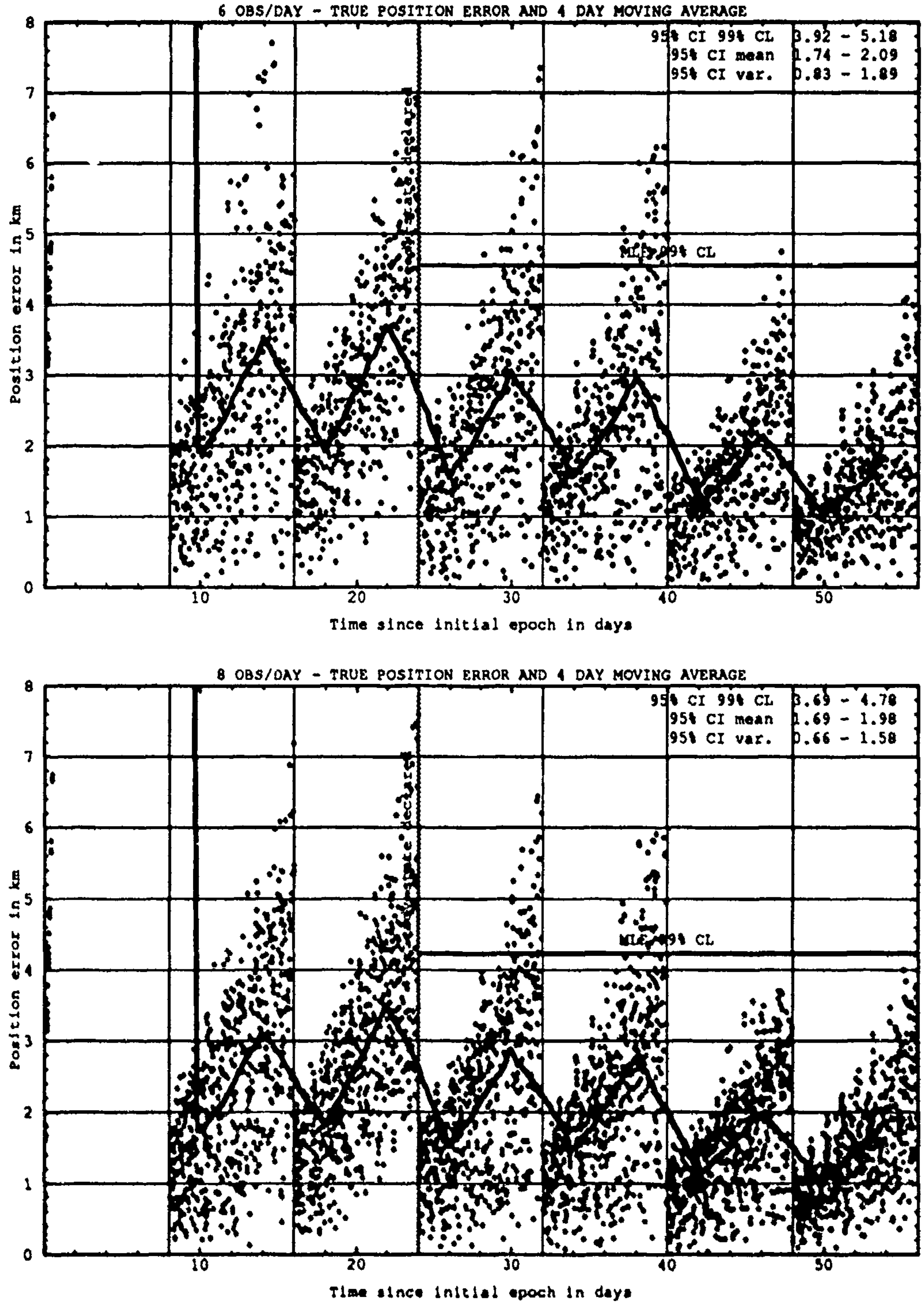


Figure H.44. Class: 2-1-4 (Catalog Number 10393), LUPI 8, OPD 6 and 8.

H.5.2 Differential Corrector Performance. The table below provides statistical information on the last-pass residuals of the differential corrector.

Table H.10. Selected Last-Pass 95% Confidence Interval Statistics.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
10393	8	8	0.56	0.59	0.08	0.09	1.21	1.30

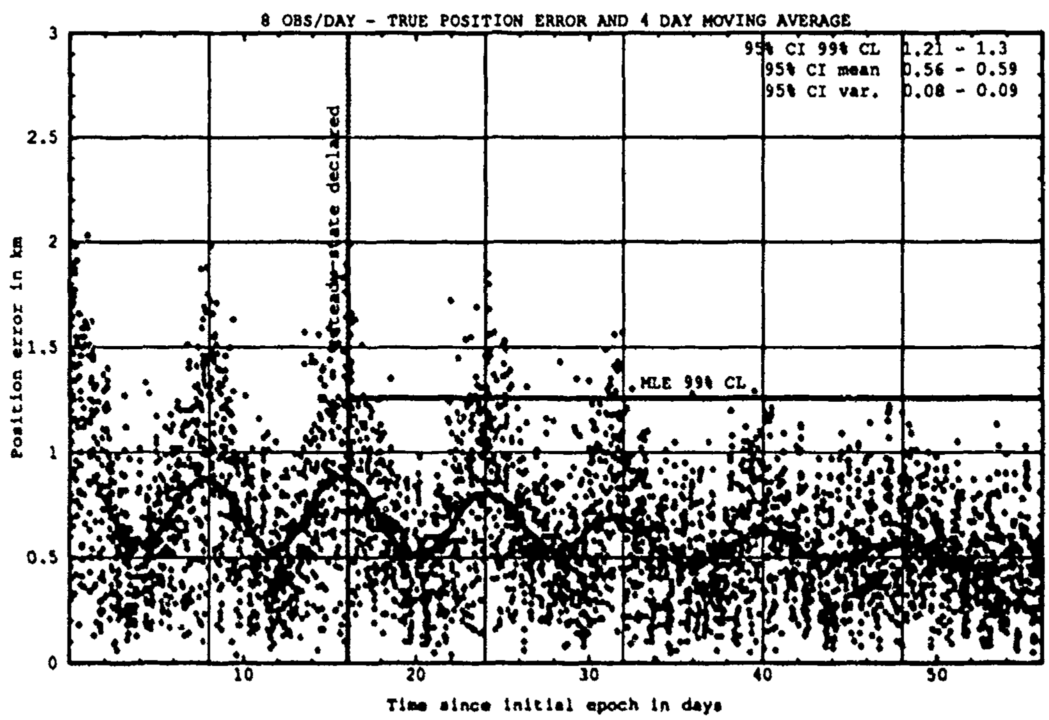


Figure H.45. Last-Pass — Class: 2-1-4 (Catalog Number 10393), LUPI 8, OPD 8.

H.6 CLASS: 2-2-3 (NORAD Catalog Number 19859)

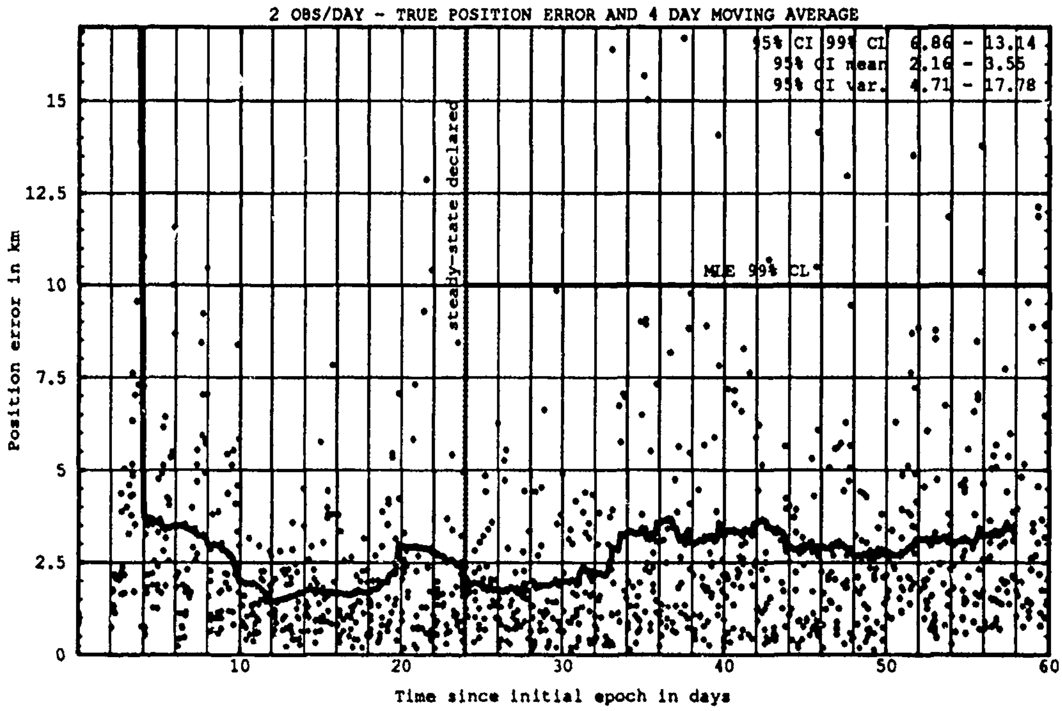
H.6.1 Prediction Performance. The tables below provide the statistical information on the ability of the estimates to predict for 1 LUPI after differential correction.

Table H.16. 95% Confidence Interval Analysis on Class: 2–2–3.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
19859	2	2	2.16	3.55	4.71	17.78	6.86	13.14
19859	2	4	1.73	2.26	1.73	2.78	4.76	6.13
19859	2	6	1.50	1.91	1.06	2.74	3.98	5.64
19859	2	8	1.48	1.85	1.07	2.26	3.95	5.26
19859	4	2	2.15	3.40	2.61	8.11	5.79	9.90
19859	4	4	1.91	2.73	2.12	5.06	5.11	7.95
19859	4	6	1.76	2.33	1.75	4.00	4.86	6.89
19859	4	8	1.64	2.07	1.25	2.81	4.26	5.89
19859	6	2	2.43	4.10	2.74	15.25	6.77	12.65
19859	6	4	2.05	2.82	2.04	5.29	5.36	8.06
19859	6	6	1.82	2.42	1.49	3.49	4.67	6.68
19859	6	8	1.74	2.30	0.96	3.16	4.17	6.26
19859	8	2	1.98	3.43	1.36	7.58	4.98	9.46
19859	8	4	2.02	3.34	1.52	5.92	5.03	8.76
19859	8	6	2.08	3.04	1.74	4.51	5.30	7.81
19859	8	8	1.87	3.05	1.10	4.60	4.70	7.72

Table H.17. ANOVA Analysis on Class: 2–2–3.

Source of Variation	Degrees of Freedom	Sum of Square	Mean Square	F	Table F
Replications	9	165.00	18.33	4.61	1.88
Main Effects:					
LUPI	3	10.59	3.53	0.8886	2.60
OPD	3	276.79	92.96	23.21	2.60
Interaction	9	87.63	9.74	2.45	1.88
Error	135	536.69	3.98		
Total	159	1076.72			



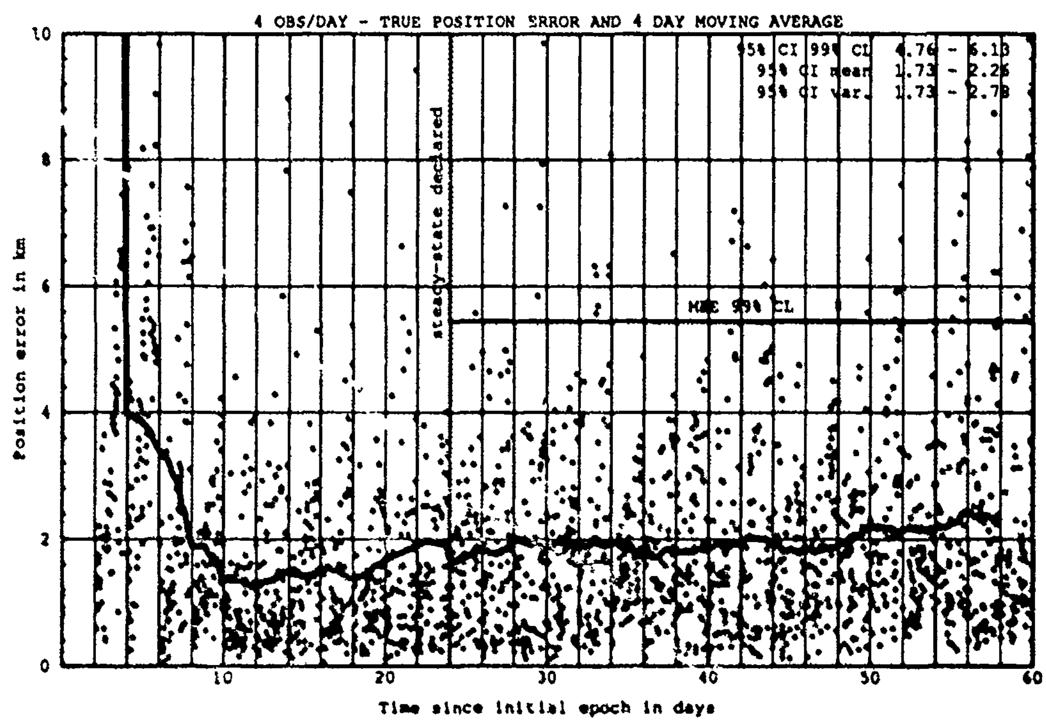
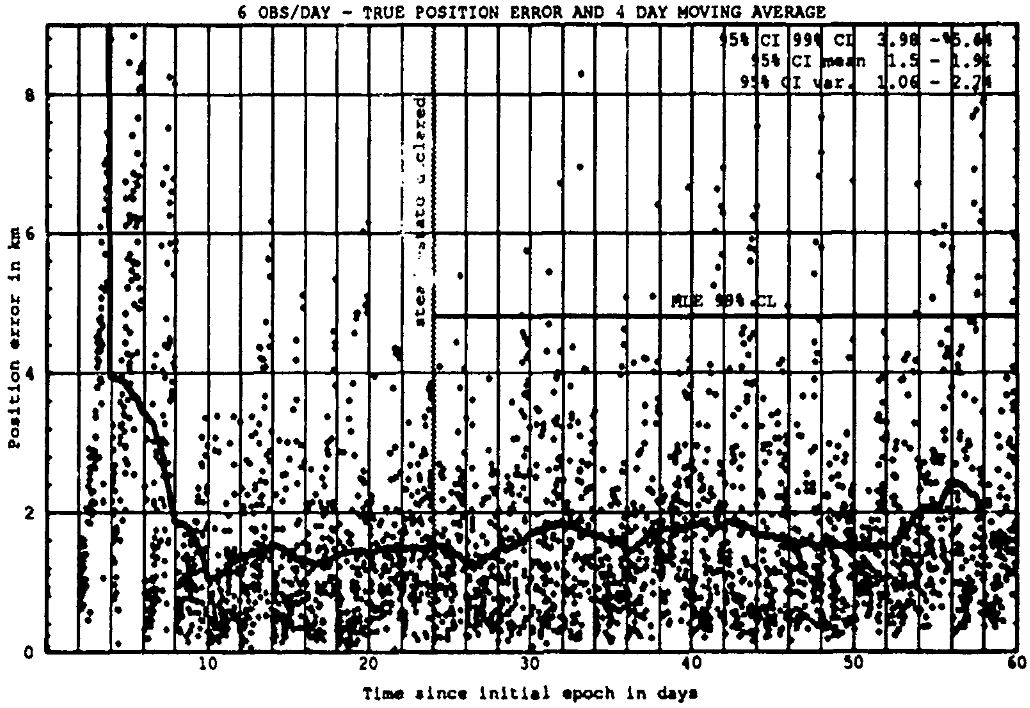


Figure H.46. Class: 2-2-3 (Catalog Number 19859), LUPI 2, OPD 2 and 4.



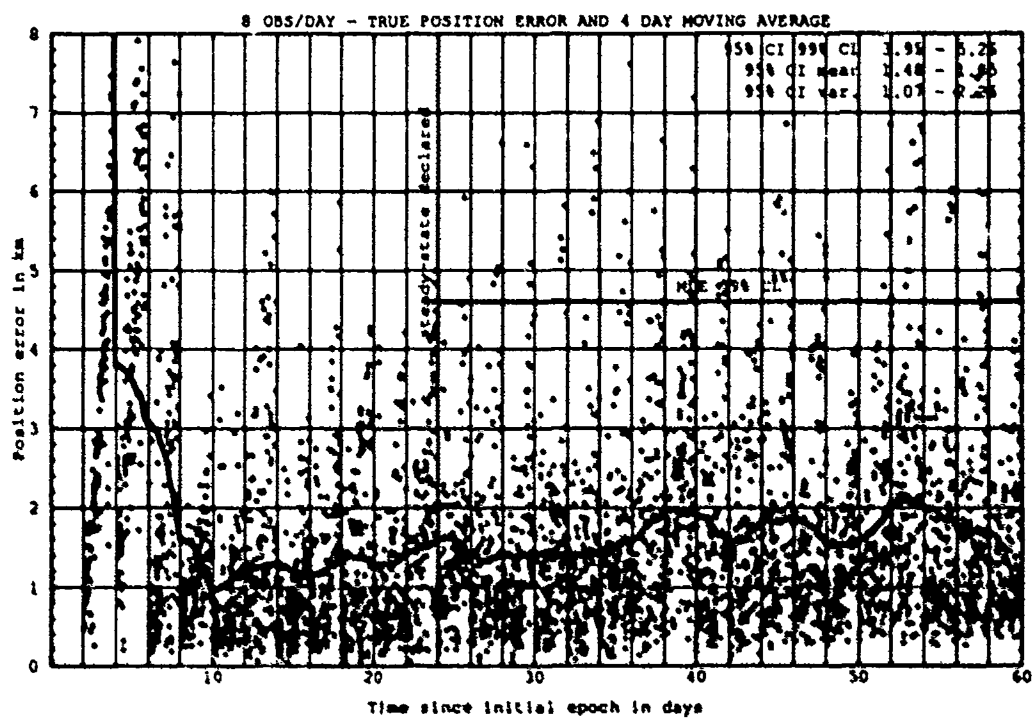
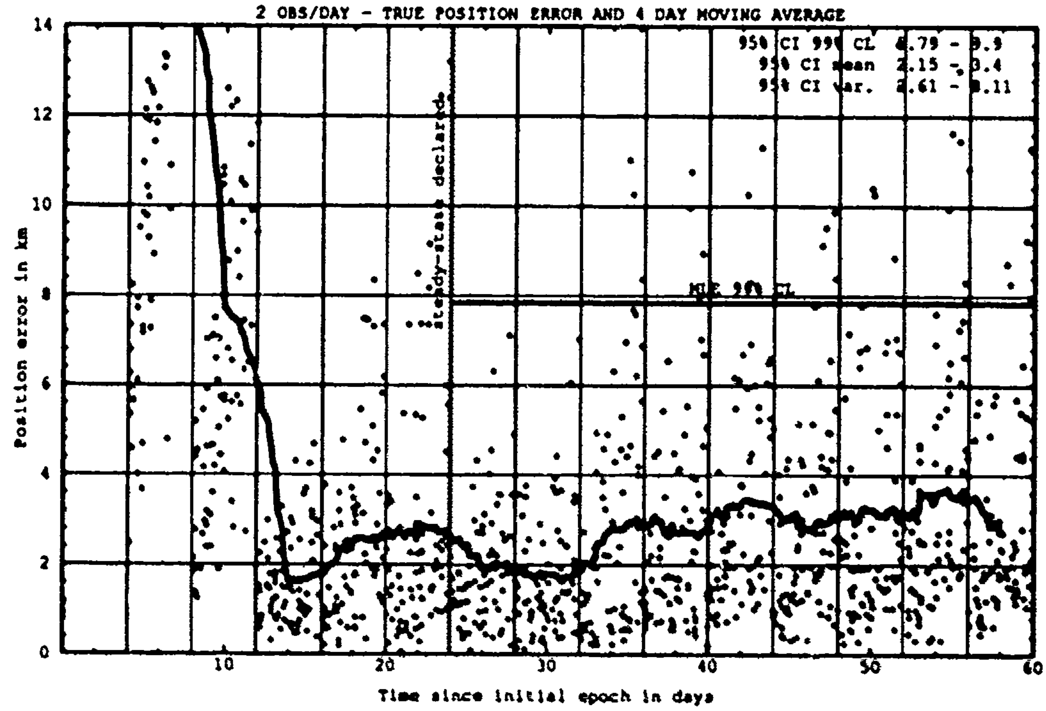


Figure II.47. Class: 2-2-3 (Catalog Number 19859), LUPI 2, OPD 6 and 8.



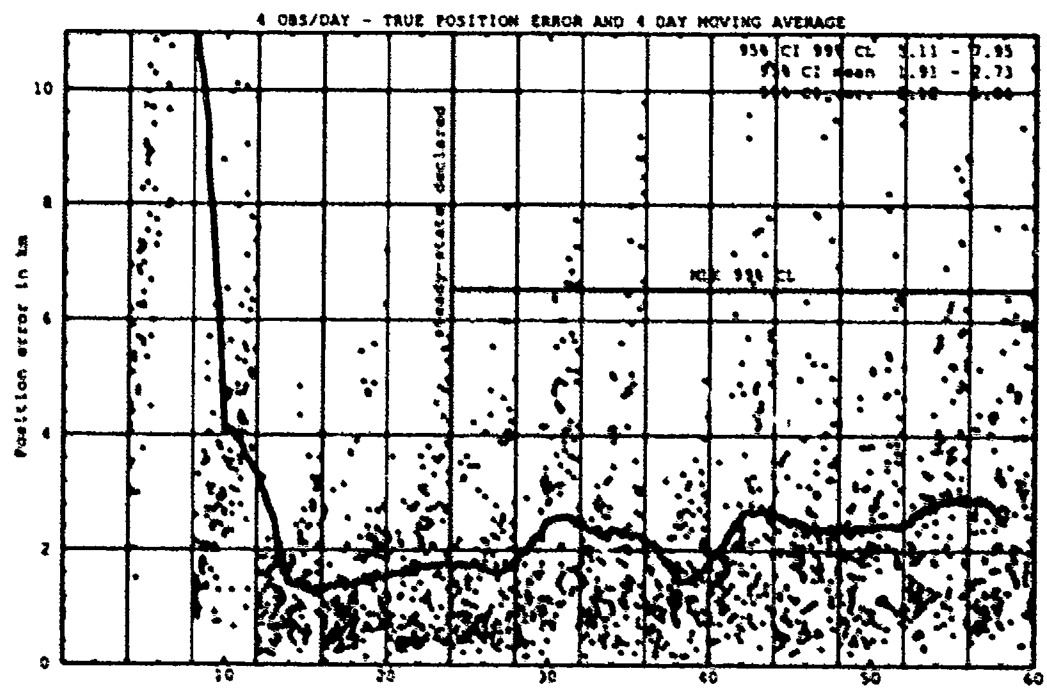
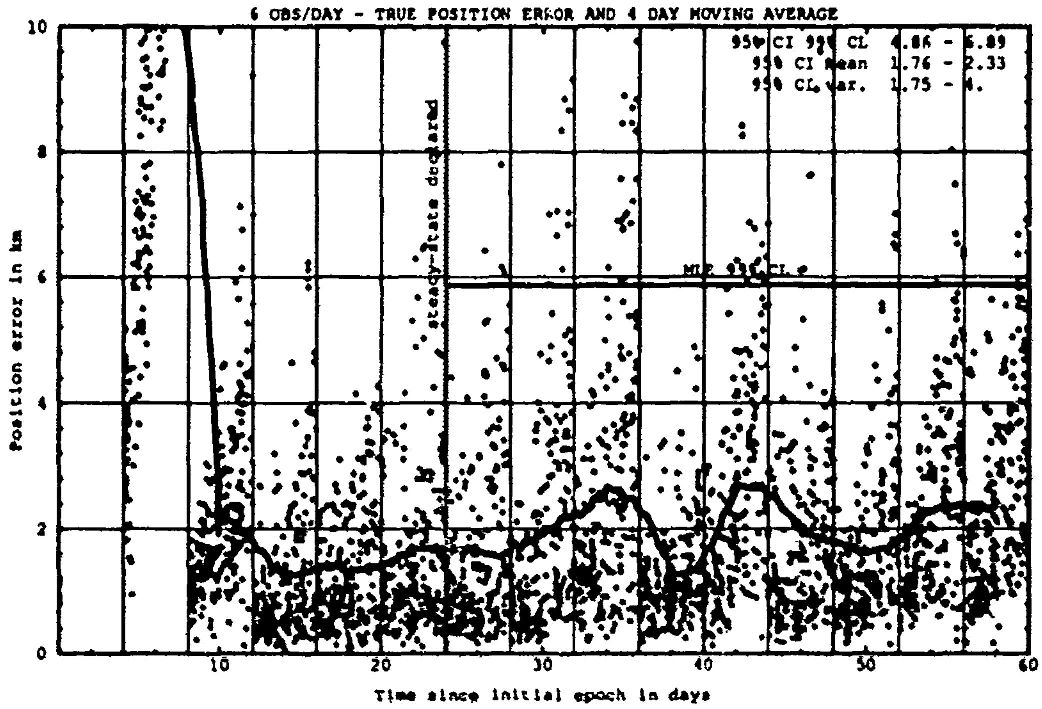


Figure H.48. Class: 2-2-3 (Catalog Number 19859), LUPI 4, OPD 2 and 4.



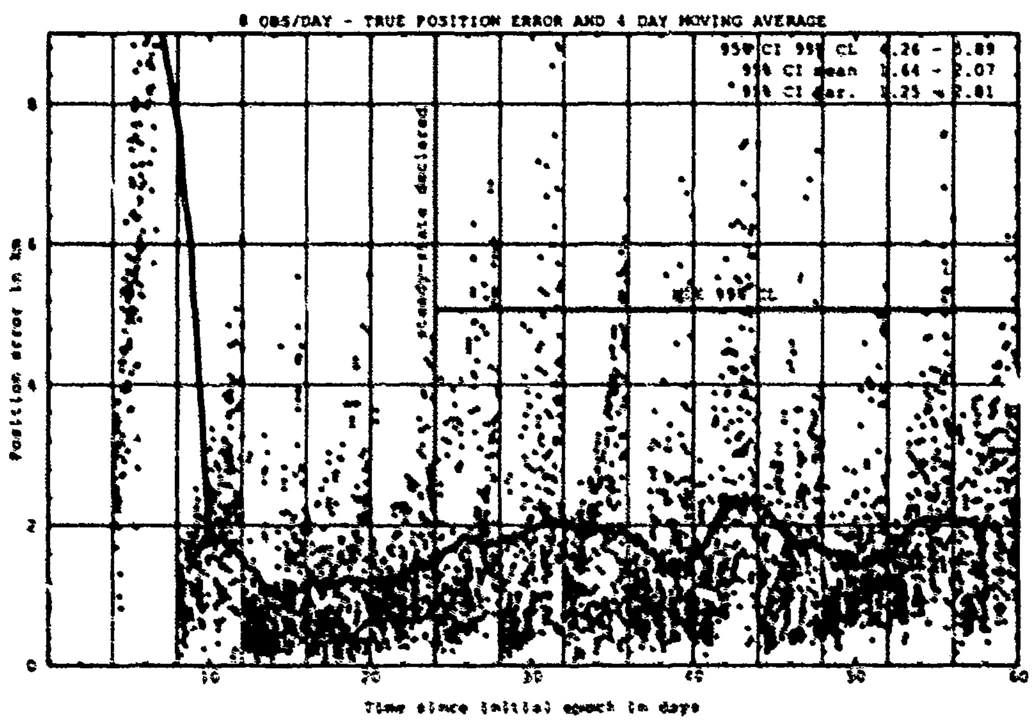


Figure H.49. Class: 2-2-3 (Catalog Number 19859), LUPI 4, OPD 6 and 8.

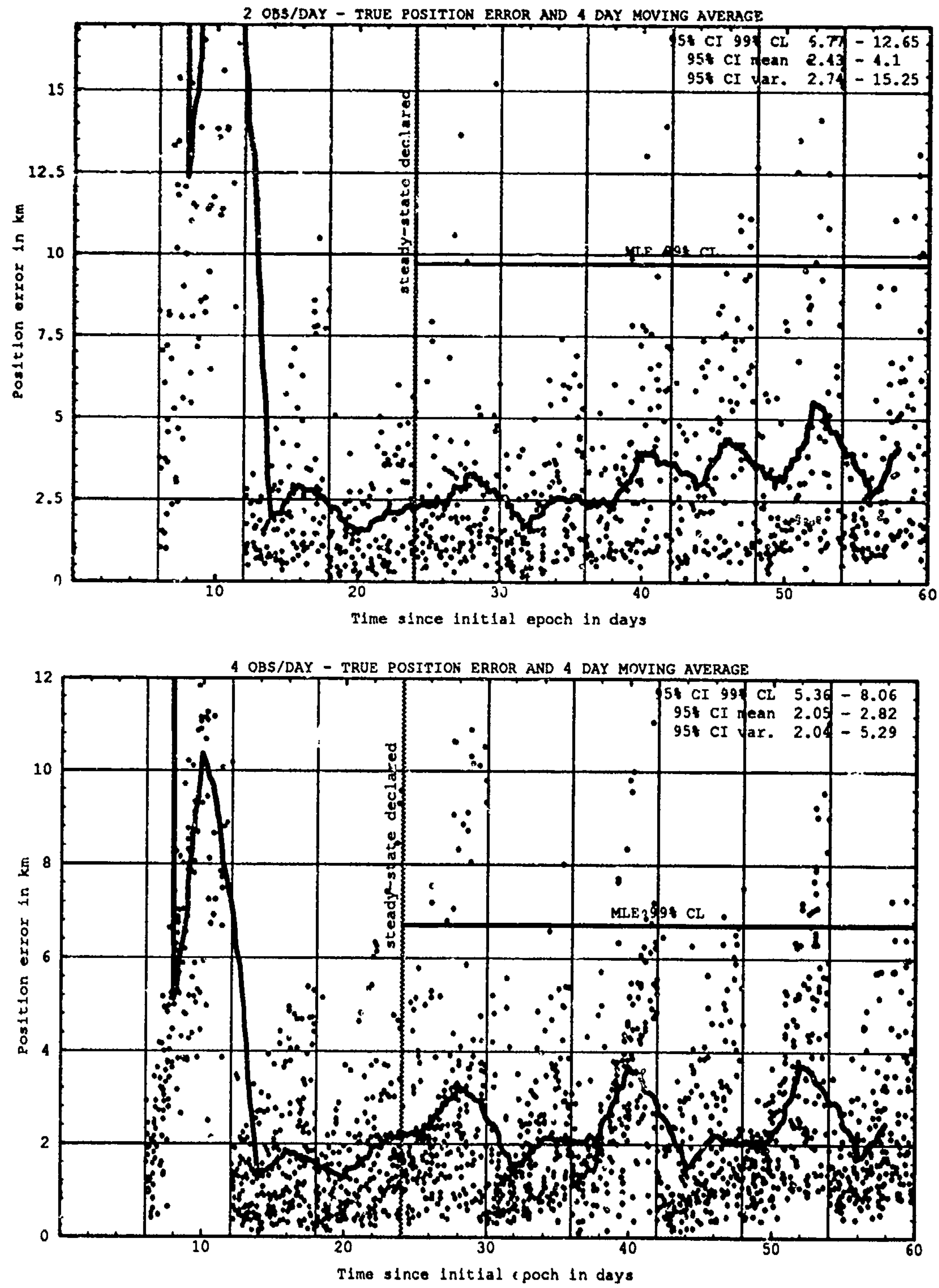
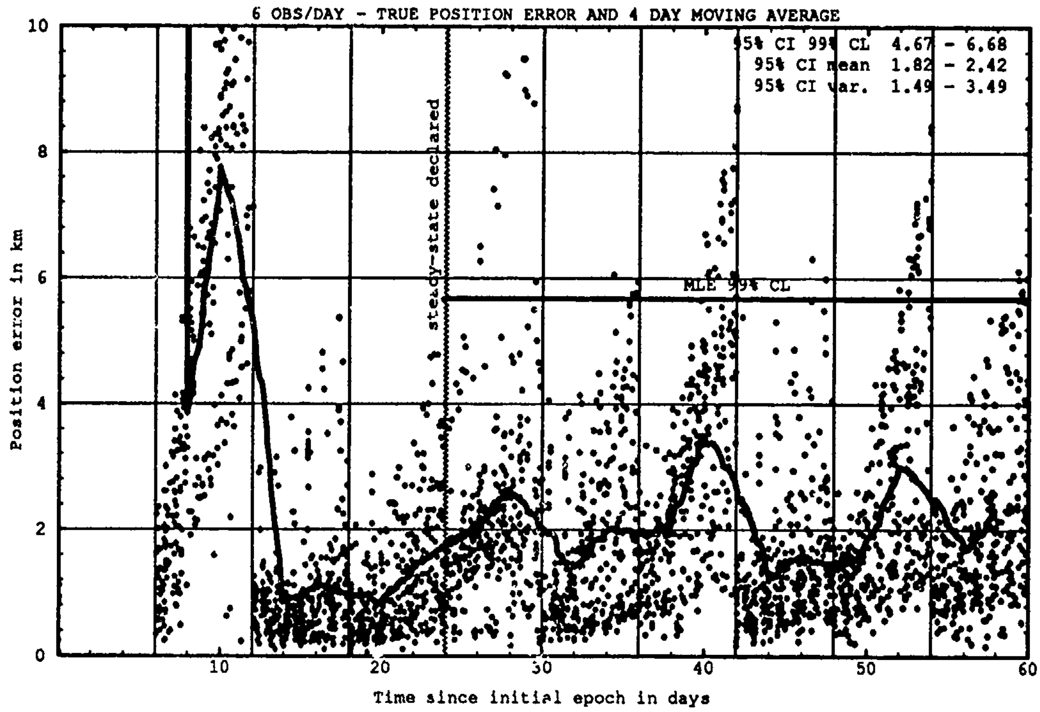


Figure H.50. Class: 2-2-3 (Catalog Number 19859), LUPI 6, OPD 2 and 4.



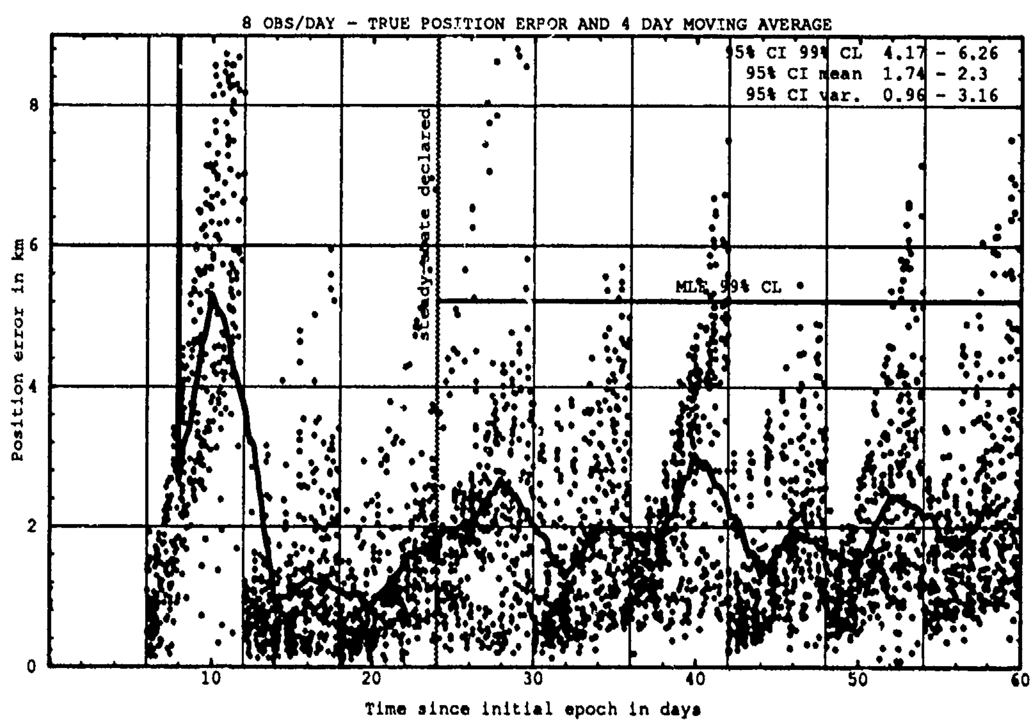
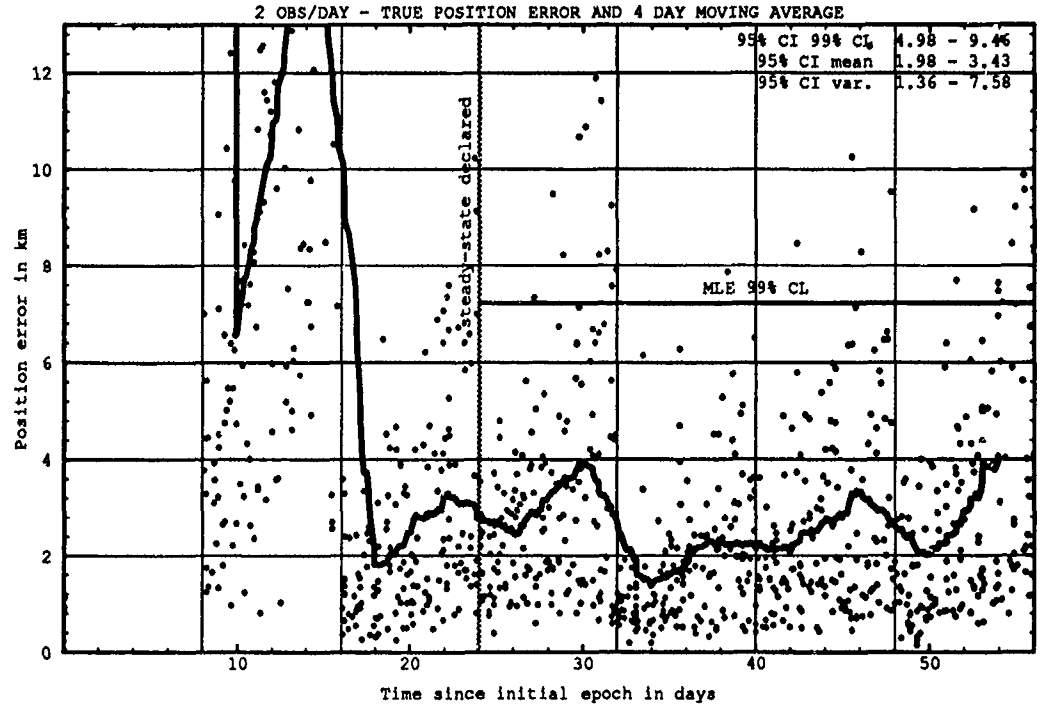


Figure H.51. Class: 2-2-3 (Catalog Number 19859), LUPI 6, OPD 6 and 8.



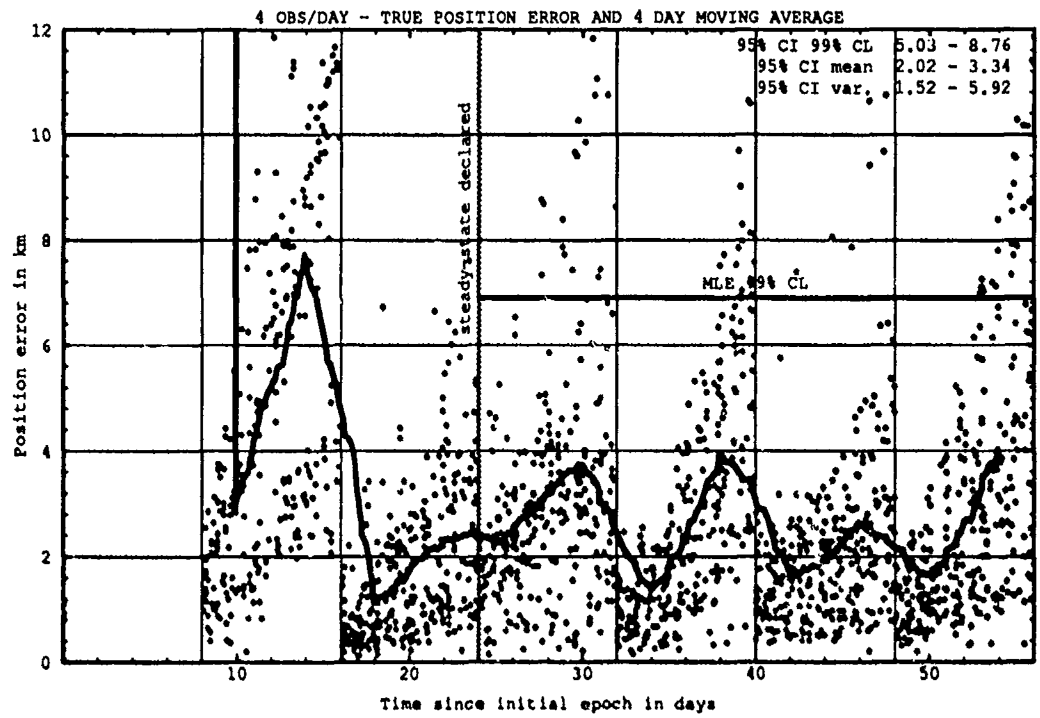


Figure H.52. Class: 2-2-3 (Catalog Number 19859), LUPI 8, OPD 2 and 4.

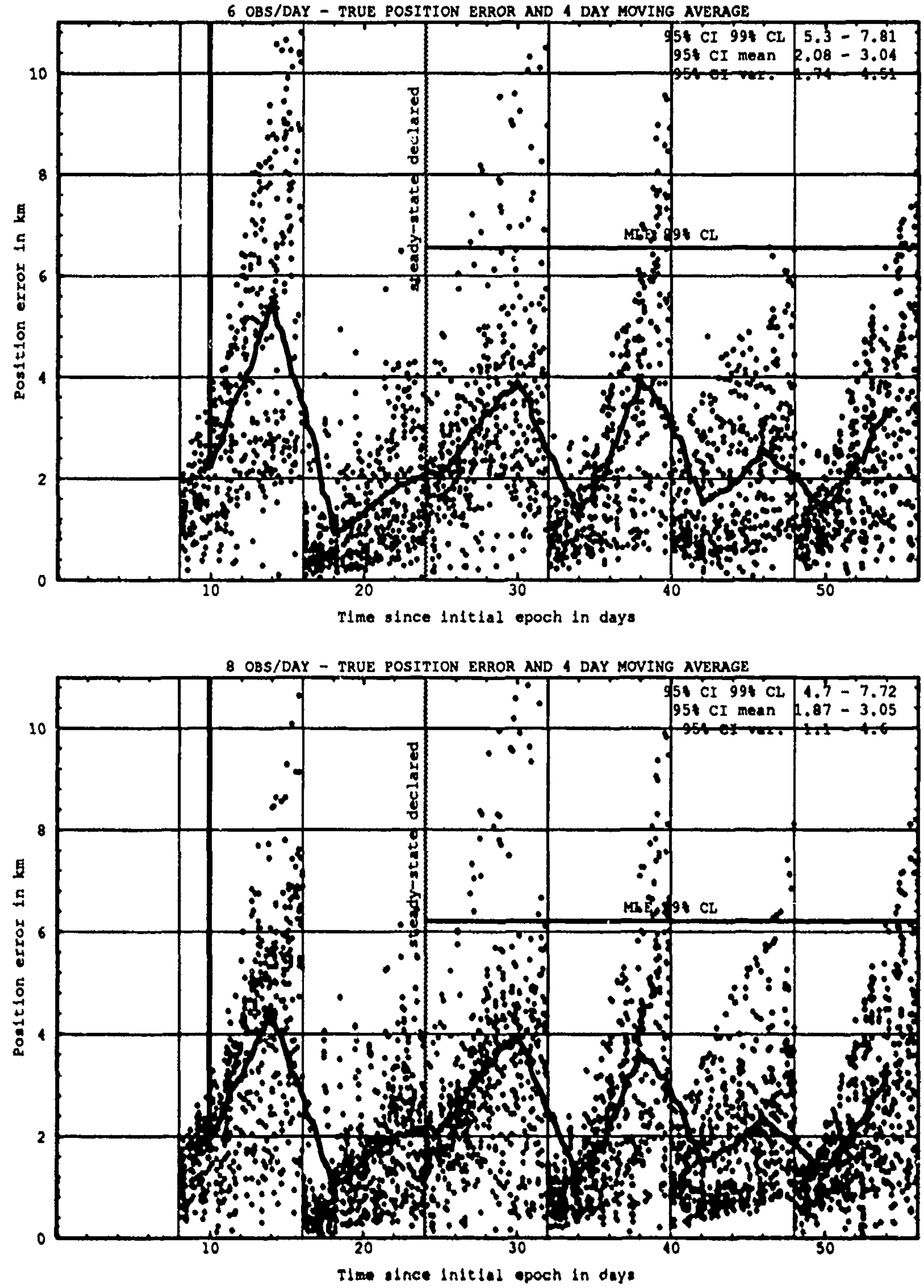


Figure H.53. Class: 2-2-3 (Catalog Number 19859), LUPI 8, OPD 6 and 8.

H.6.2 Differential Corrector Performance. **The table below provides statistical information on the last-pass residuals of the differential corrector.**

Table H.12. Selected Last-Pass 95% Confidence Interval Statistics.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
19859	8	8	0.72	0.79	0.13	0.19	1.57	1.78

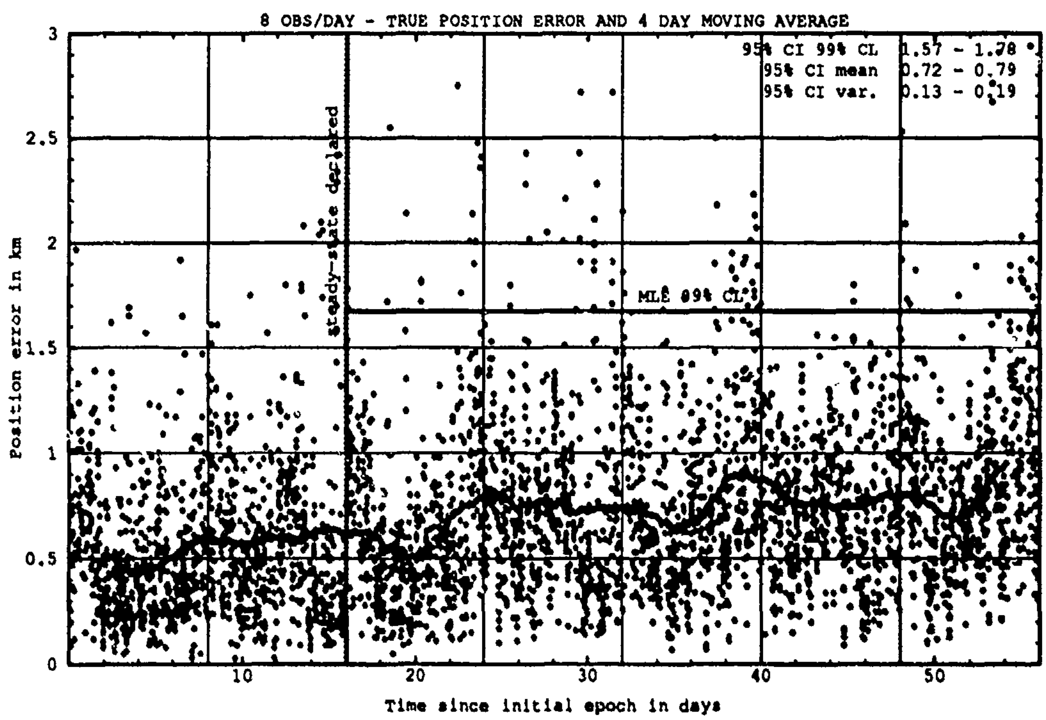


Figure H.54. Last-Pass — Class: 2-2-3 (Catalog Number 19859), LUPI 8, OPD 8.

H.7 CLASS: 3-1-1 (NORAD Catalog Number 01996)

H.7.1 Prediction Performance. The tables below provide the statistical information on the ability of the estimates to predict for 1 LUPI after differential correction.

Table H.19. 95% Confidence Interval Analysis on Class: 3-1-1.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
01996	2	2	3.59	4.67	6.80	18.02	9.94	14.17
01996	2	4	2.88	3.27	4.28	8.60	7.74	9.97
01996	2	6	2.61	2.98	3.45	5.96	6.95	8.60
01996	2	8	2.75	3.26	3.96	6.84	7.36	9.30
01996	4	2	9.40	12.89	49.48	122.80	26.11	37.87
01996	4	4	9.45	11.22	43.61	73.54	25.10	30.77
01996	4	6	9.04	10.38	36.56	62.17	23.32	28.40
01996	4	8	8.89	9.83	39.57	50.33	23.80	26.02
01996	6	2	22.49	25.36	243.10	342.69	58.99	68.05
01996	6	4	19.44	21.38	178.25	203.37	50.67	54.37
01996	6	6	18.16	19.63	145.37	177.94	46.40	50.43
01996	6	8	17.76	19.50	138.18	168.70	45.17	49.61
01996	8	2	33.68	43.85	466.65	918.71	84.95	112.91
01996	8	4	33.79	38.83	502.01	637.48	86.04	97.31
01996	8	6	33.27	36.84	484.39	606.50	84.49	93.99
01996	8	8	32.70	35.04	452.60	556.18	82.31	89.70

Table H.20. ANOVA Analysis on Class: 3-1-1.

Source of Variation	Degrees of Freedom	Sum of Square	Mean Square	F	Table F
Replications	9	1224.52	136.06	4.20	1.88
Main Effects:					
LUPI	3	151974.00	50658.00	1563.90	2.60
OPD	3	2373.27	791.09	24.42	2.60
Interaction	9	570.47	63.39	1.96	1.88
Error	135	4372.93	32.39		
Total	159	160515.21			

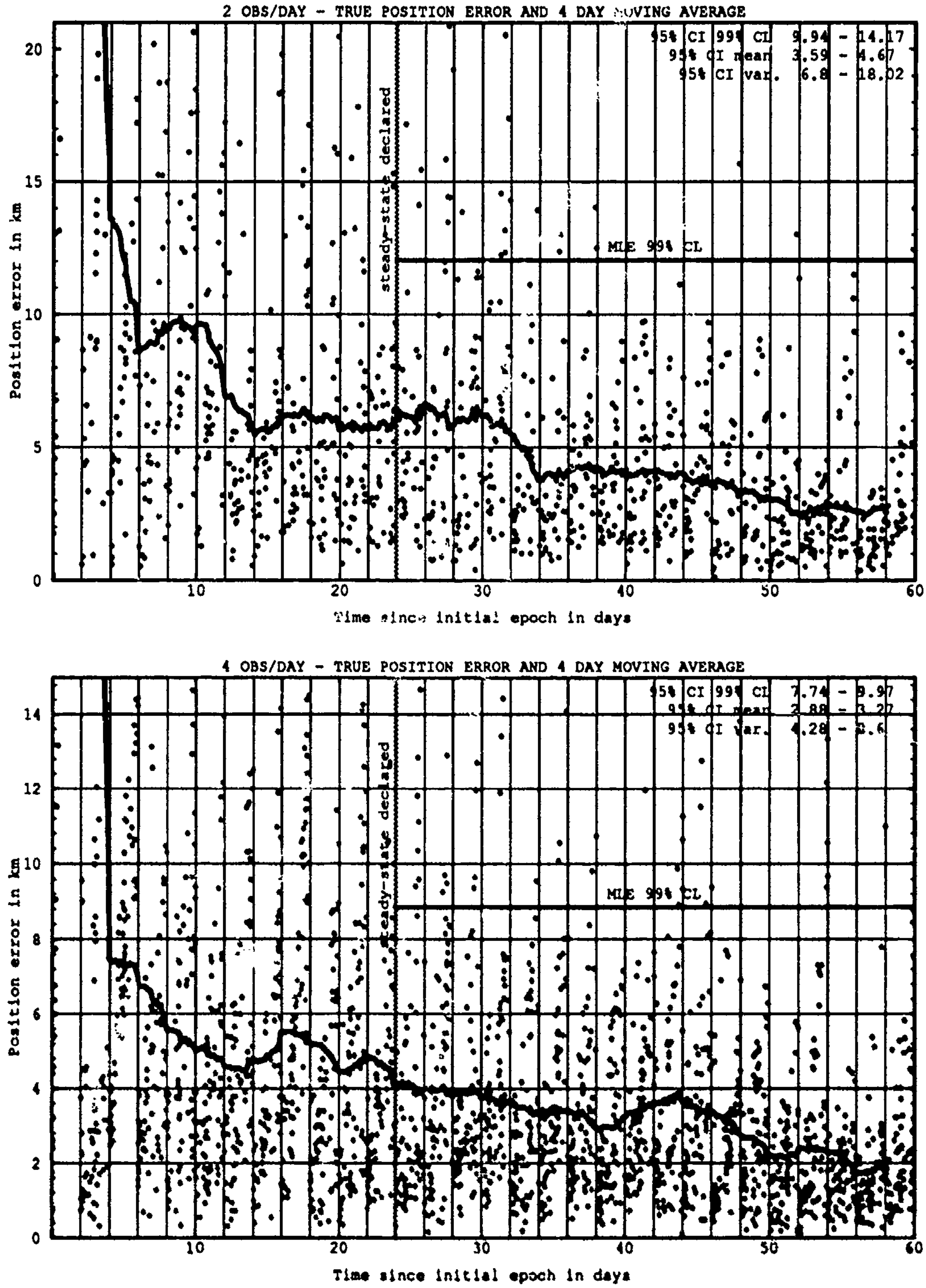


Figure H.55. Class: 3-1-1 (Catalog Number 01996), LUPI 2, OPD 2 and 4.

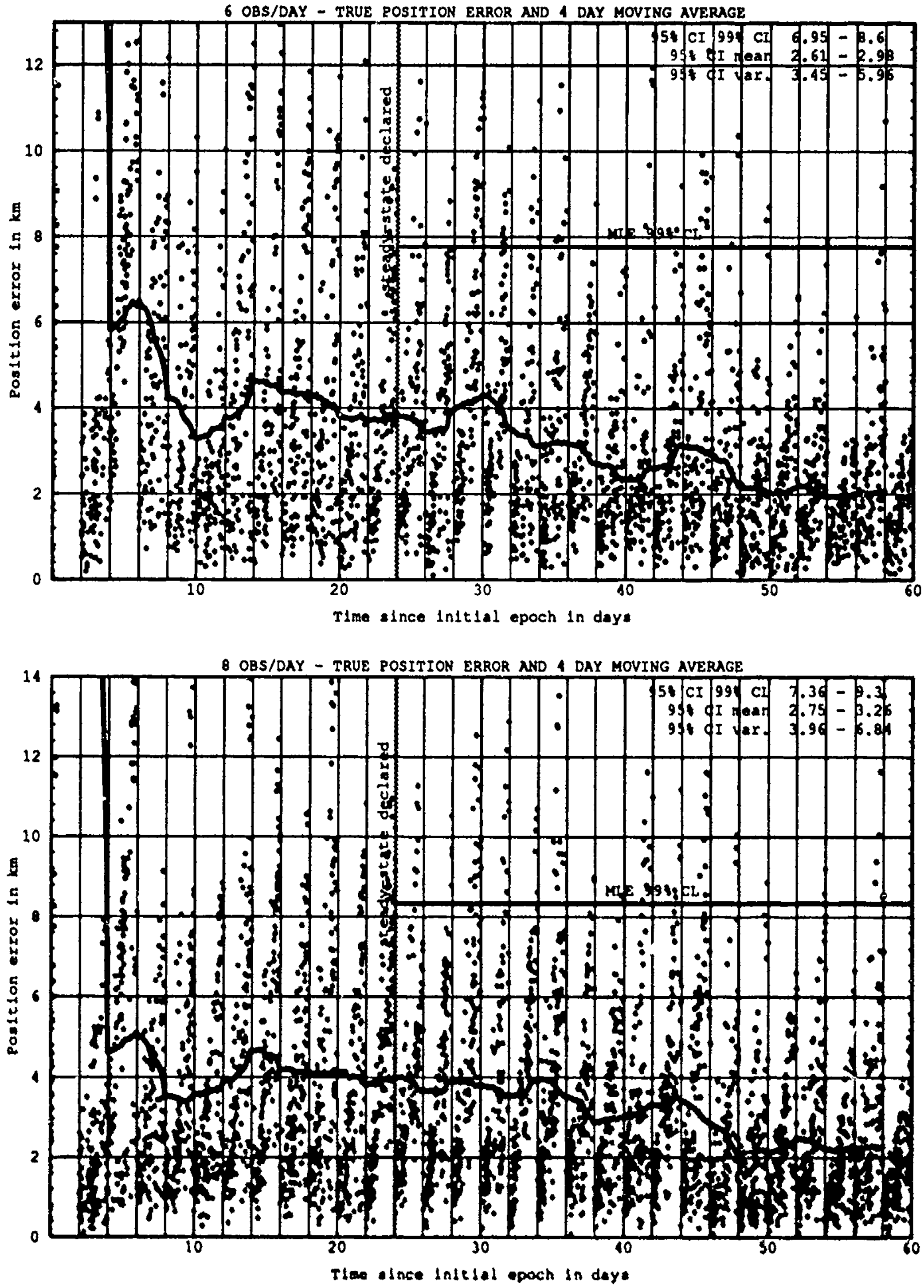


Figure H.56. Class: 3-1-1 (Catalog Number 01996), LUPI 2, OPD 6 and 8.

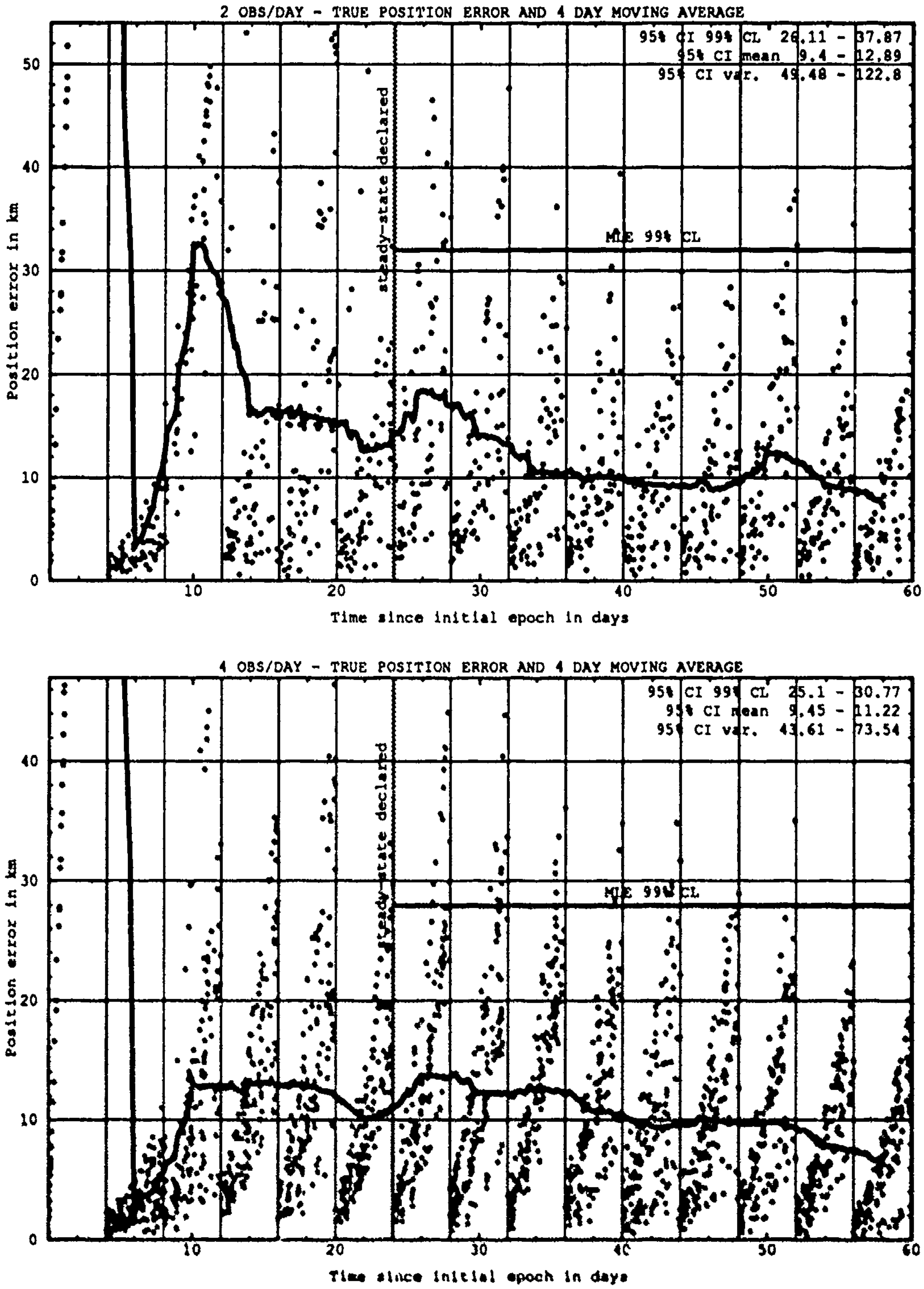
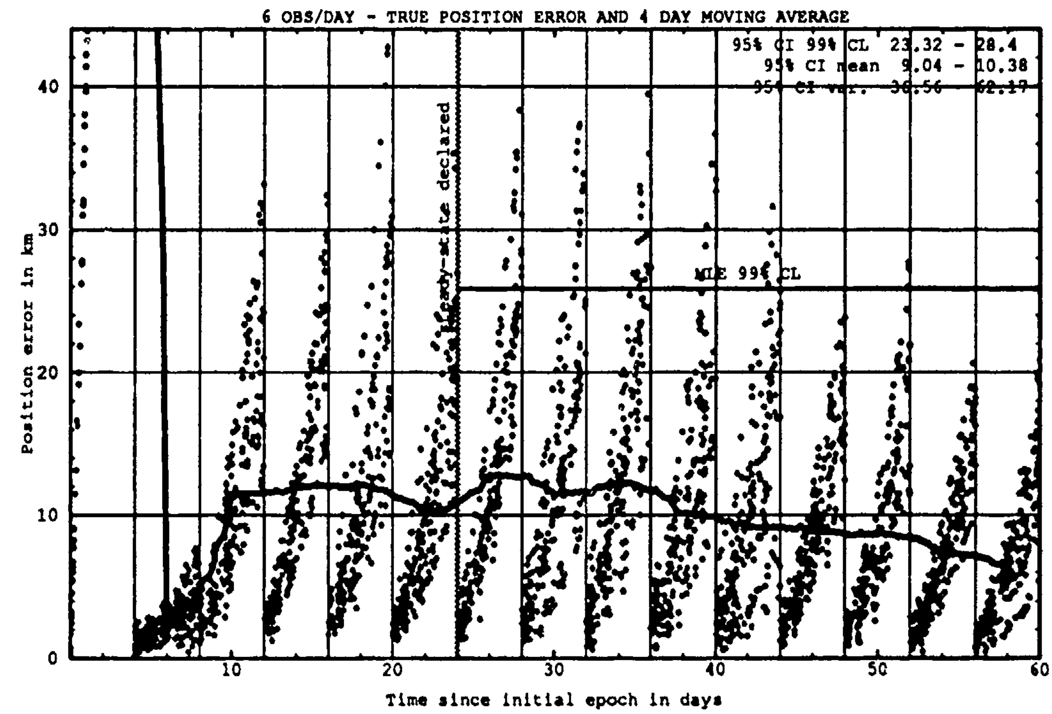


Figure H.57. Class: 3-1-1 (Catalog Number 01996), LUPI 4, OPD 2 and 4.



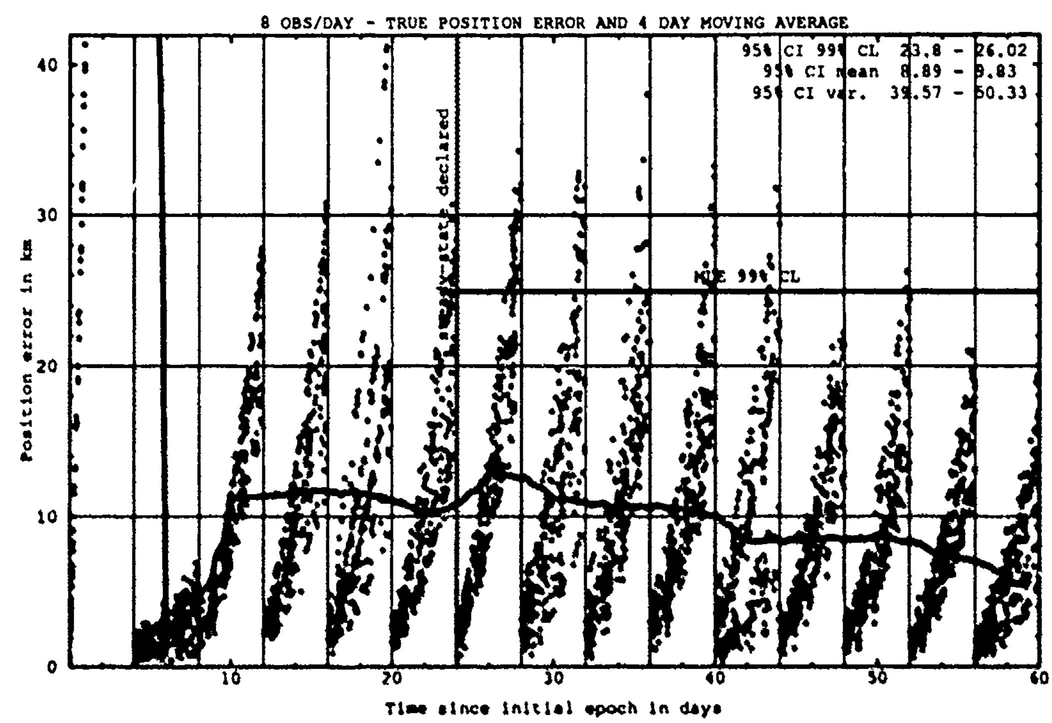
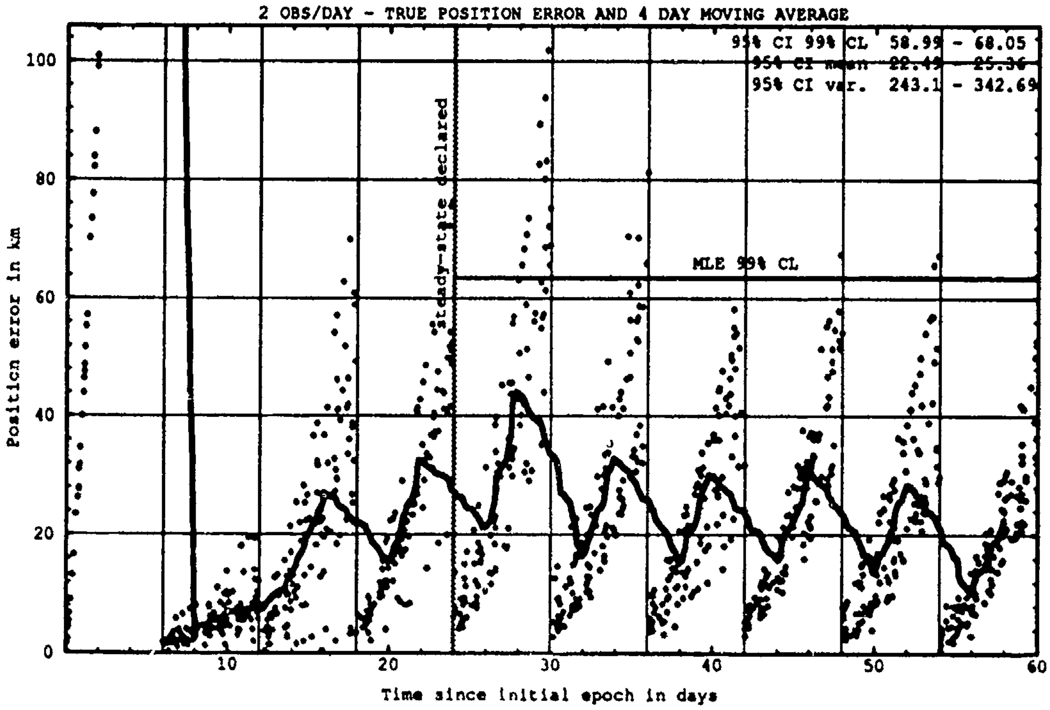


Figure H.58. Class: 3-1-1 (Catalog Number 01996), LUPI 4, OPD 6 and 8.



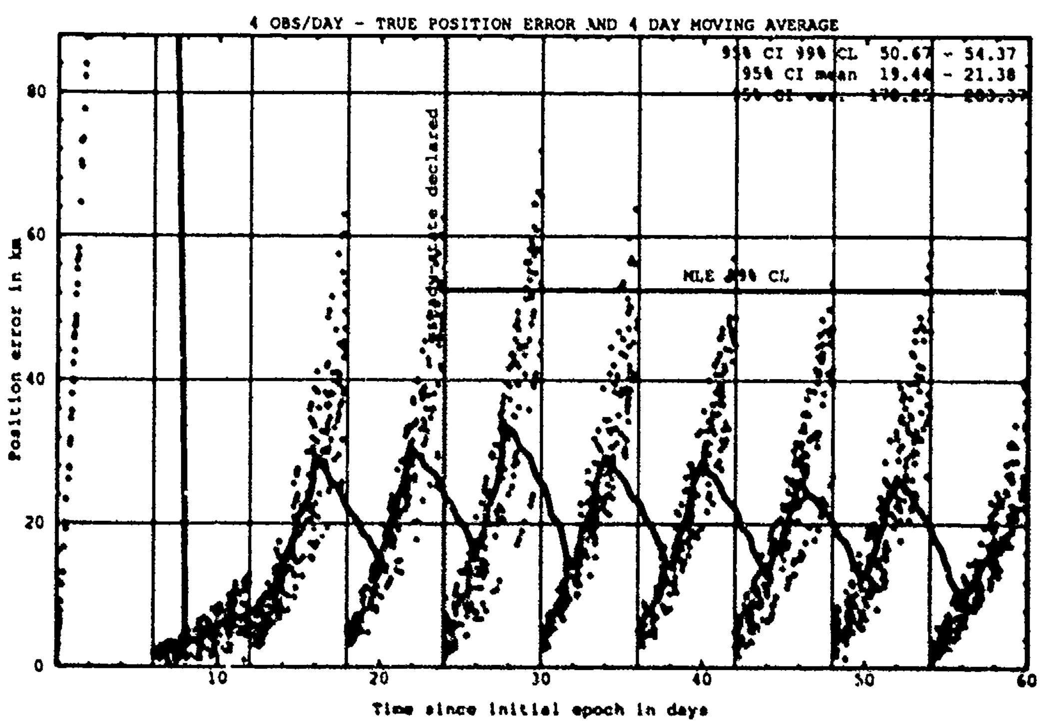


Figure H.59. Class: 3-1-1 (Catalog Number 01996), LUPI 6, OPD 2 and 4.

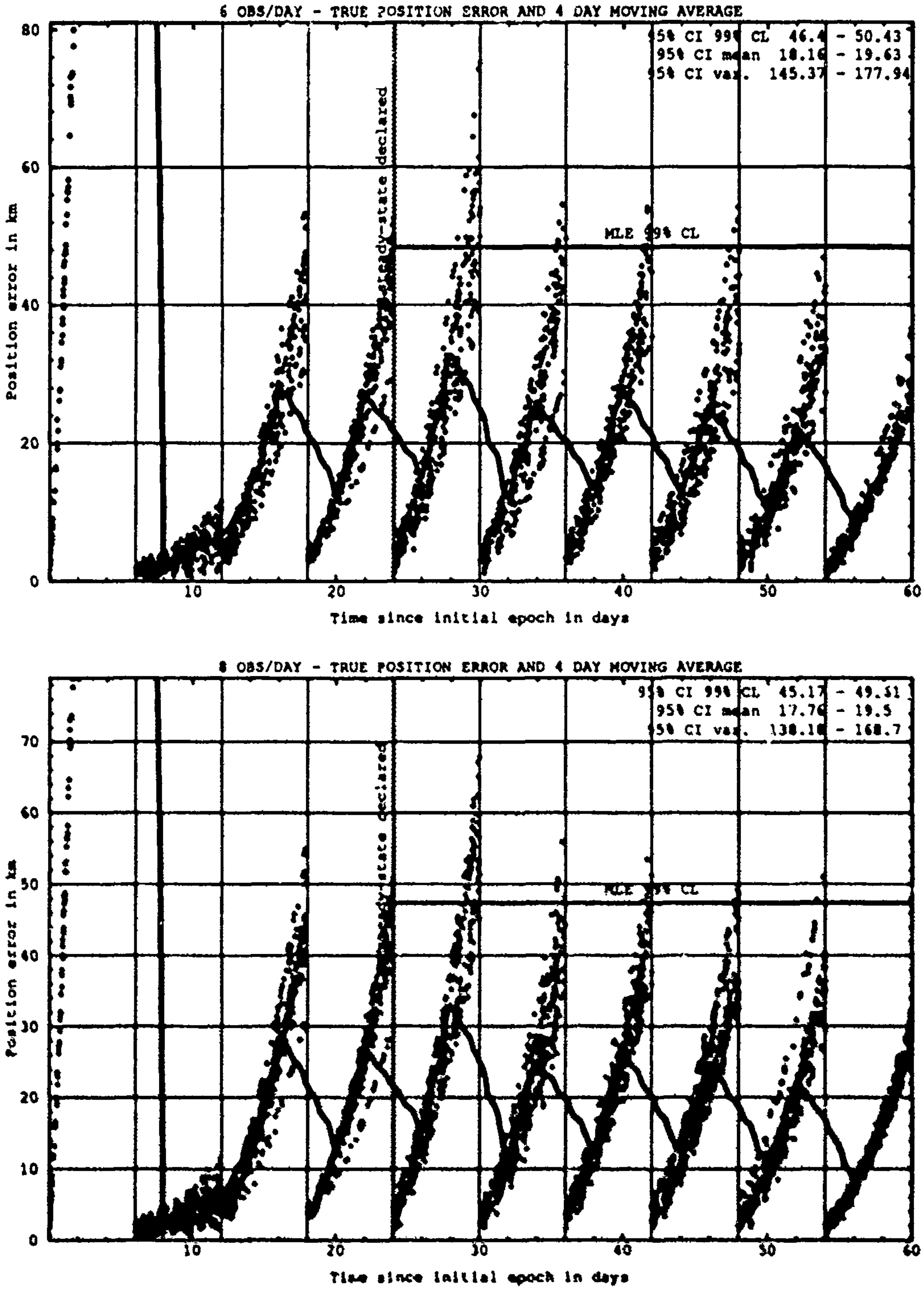
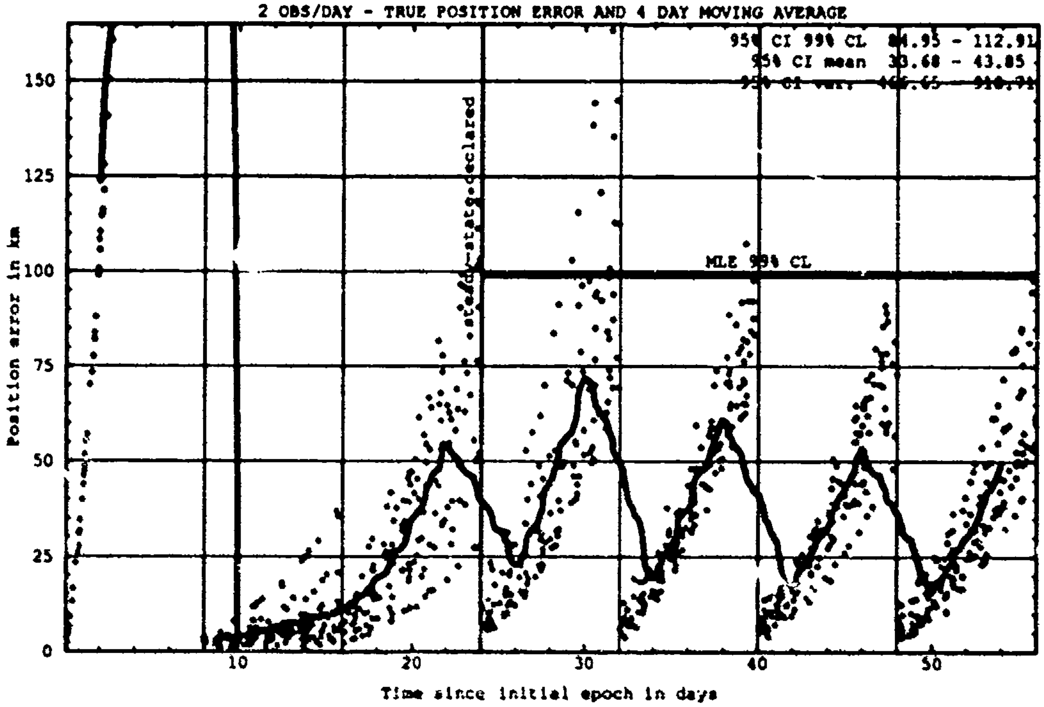


Figure H.60. Class: 3-1-1 (Catalog Number 01996), LUPI 6, OPD 6 and 8.



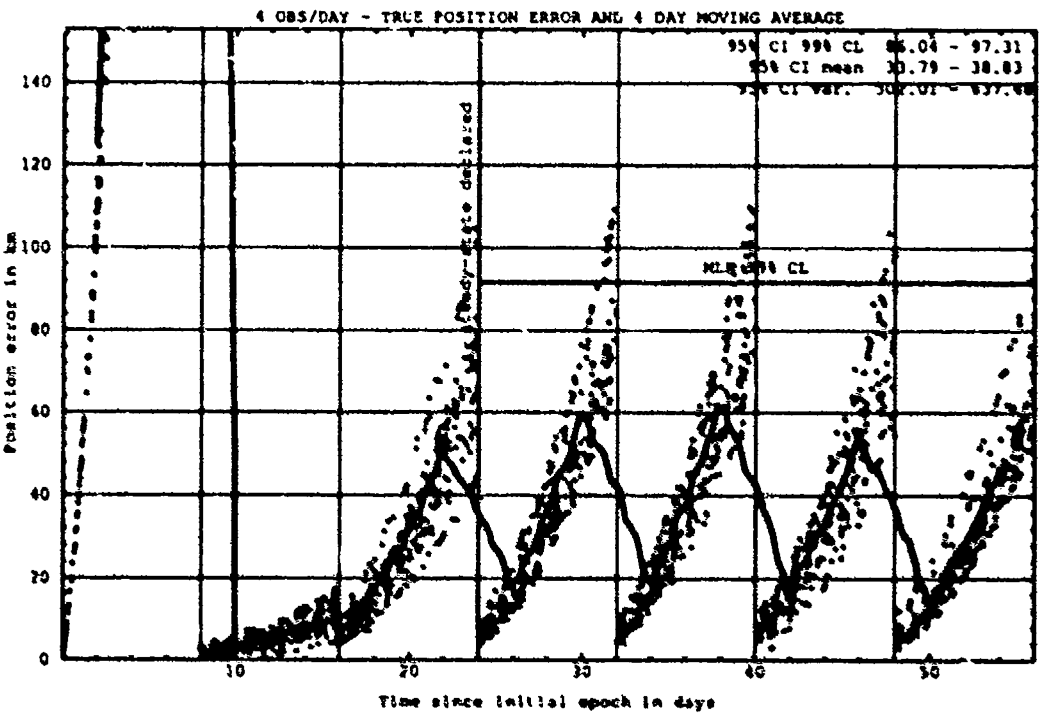


Figure H.61. Class: 3-1-1 (Catalog Number 01996), LUPI 8, OPD 2 and 4.

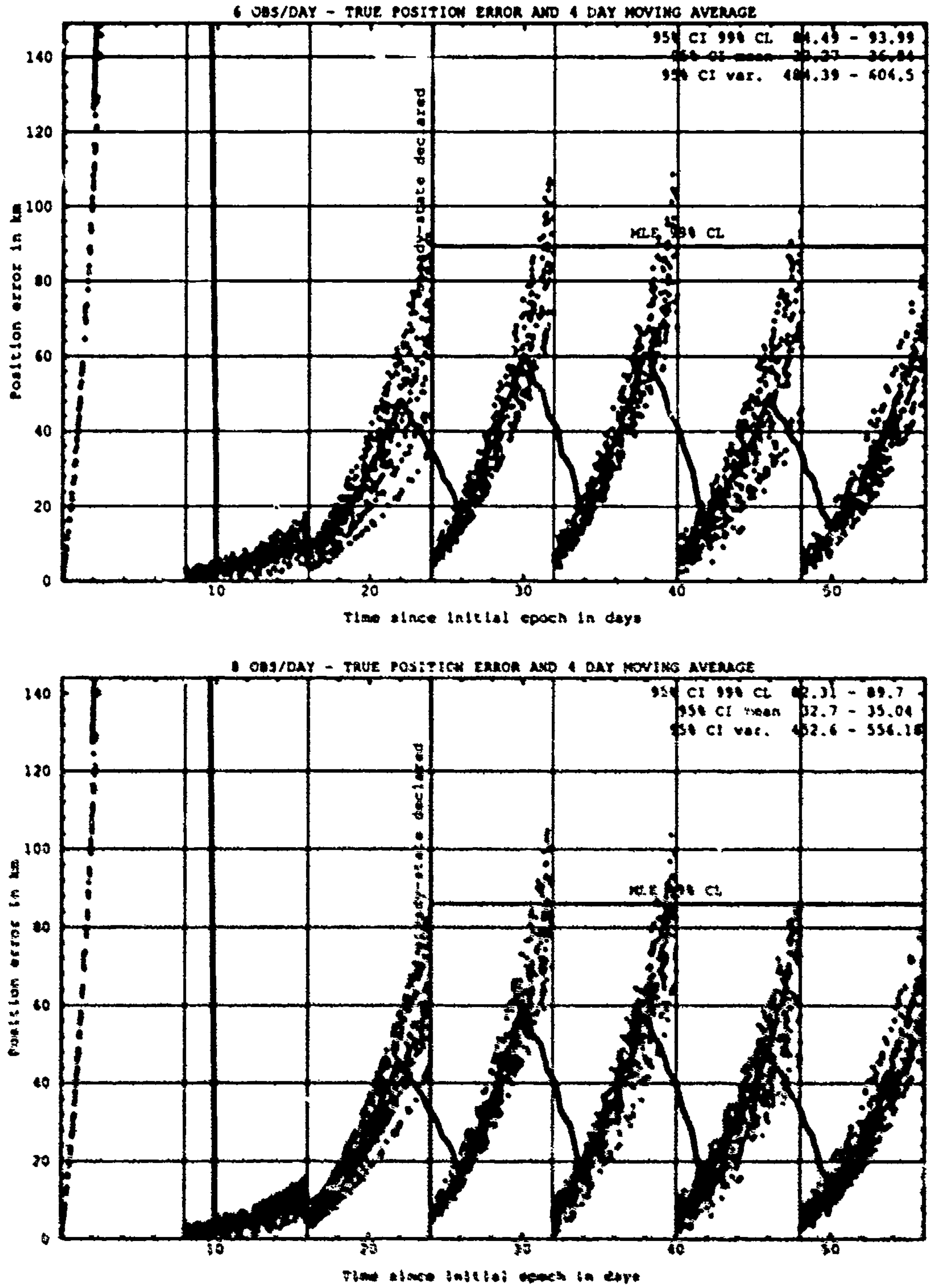


Figure H.62. Class: 3-1-1 (Catalog Number 01996), LUPI 8, OPD 6 and 8.

H.7.2 Differential Corrector Performance. The table below provides statistical information on the last-pass residuals of the differential corrector.

Table H.14. Selected Last-Pass 95% Confidence Interval Statistics.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
01996	8	8	2.35	2.58	1.38	1.88	5.09	5.75

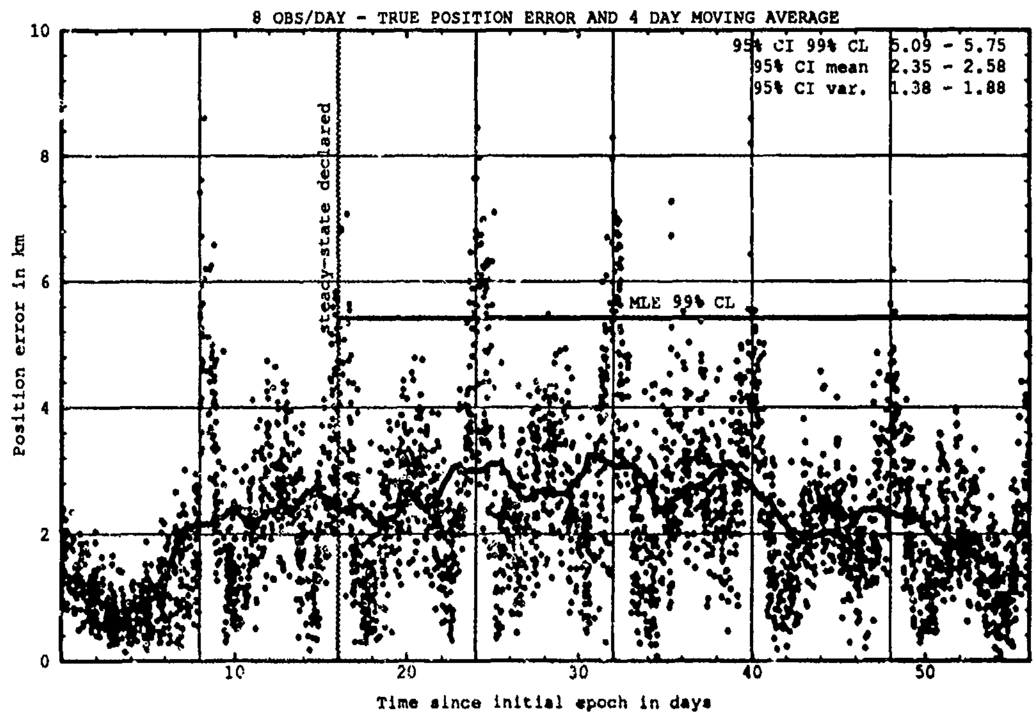


Figure H.63. Last-Pass — Class: 3-1-1 (Catalog Number 01996), LUPI 8, OPD 8.

H.8 CLASS: 3-1-2 (NORAD Catalog Number 14443)

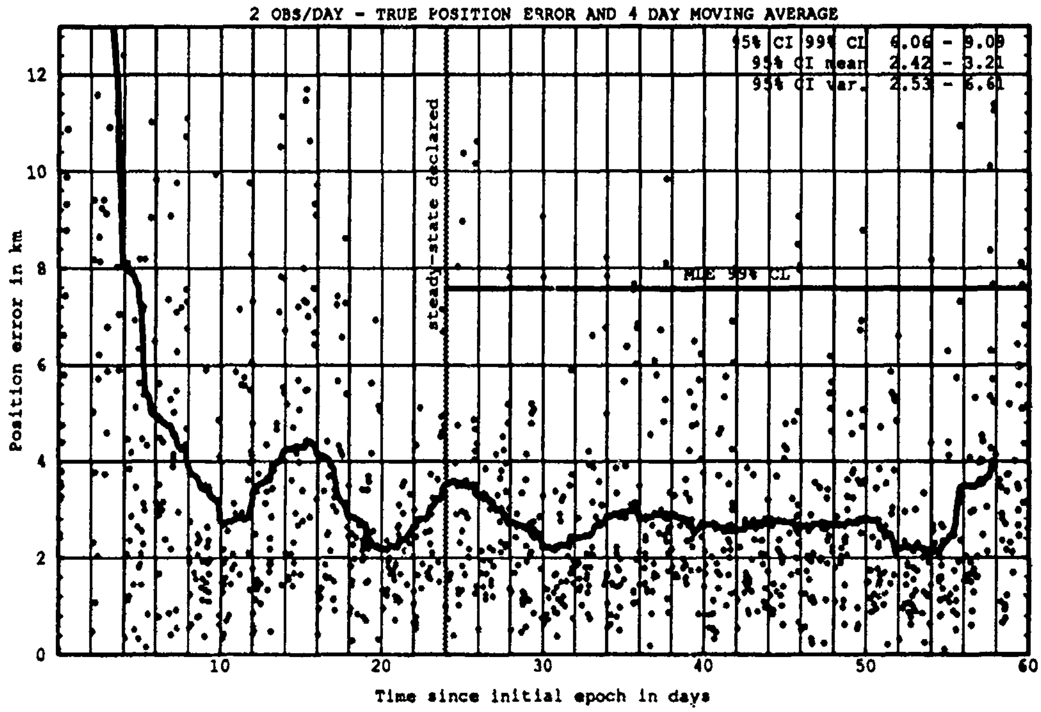
H.8.1 Prediction Performance. The tables below provide the statistical information on the ability of the estimates to predict for 1 LUPI after differential correction.

Table H.22. 95% Confidence Interval Analysis on Class: 3-1-2.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
14443	2	2	2.42	3.21	2.53	6.61	6.06	9.09
14443	2	4	1.98	2.20	1.86	2.36	5.24	5.68
14443	2	6	1.81	2.07	1.28	2.31	4.46	5.54
14443	2	8	1.62	1.92	1.22	2.00	4.20	5.17
14443	4	2	3.24	4.30	5.10	9.88	8.34	11.60
14443	4	4	2.57	3.71	2.83	7.92	6.63	10.03
14443	4	6	2.23	3.29	1.80	6.34	5.52	8.90
14443	4	8	2.16	2.80	2.09	4.29	5.52	7.56
14443	6	2	4.09	5.99	8.08	31.08	10.81	18.46
14443	6	4	3.59	4.64	5.24	15.14	9.15	13.36
14443	6	6	3.36	4.10	5.53	10.63	8.81	11.58
14443	6	8	3.36	3.85	5.80	11.17	8.98	11.51
14443	8	2	5.27	7.36	14.64	30.84	14.31	19.97
14443	8	4	5.21	6.71	11.55	21.15	13.11	17.29
14443	8	6	4.72	5.80	8.87	15.54	11.66	14.87
14443	8	8	4.72	5.57	9.18	14.01	11.80	14.20

Table H.23. ANOVA Analysis on Class: 3-1-2.

Source of Variation	Degrees of Freedom	Sum of Square	Mean Square	F	Table F
Replications	9	149.51	16.61	3.39	1.88
Main Effects:					
LUPI	3	1869.48	623.16	127.07	2.60
OPD	3	340.80	113.60	23.17	2.60
Interaction	9	20.64	2.29	0.4676	1.88
Error	135	662.03	4.90		
Total	159	3042.47			



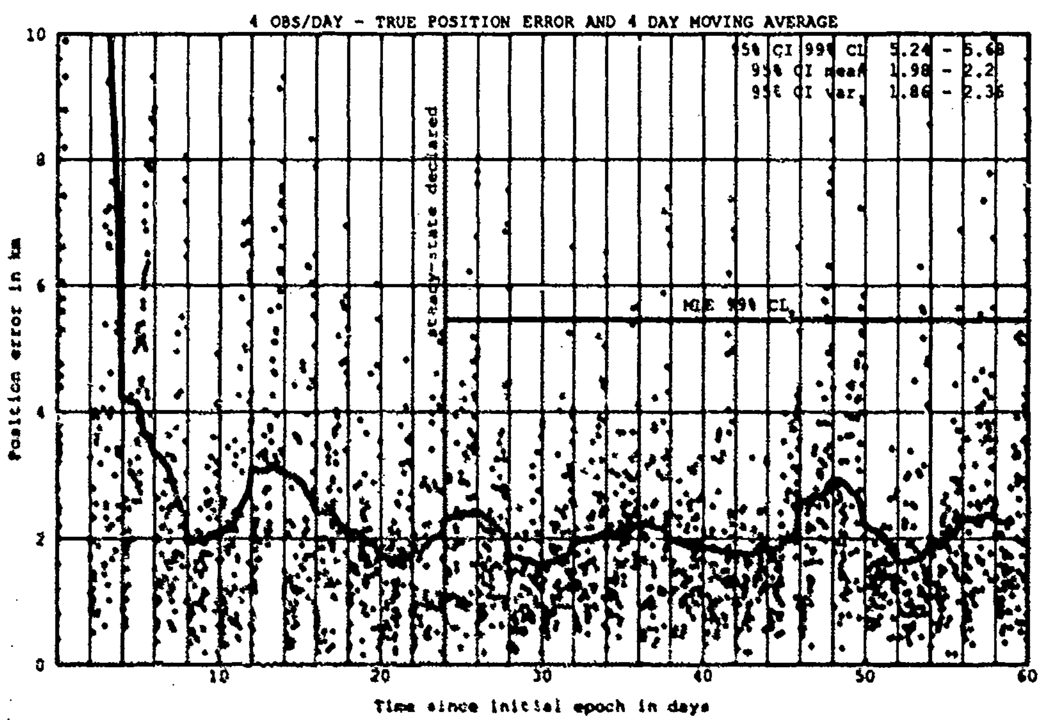


Figure II.64. Class: 3-1-2 (Catalog Number 14443), LUPI 2, OPD 2 and 4.

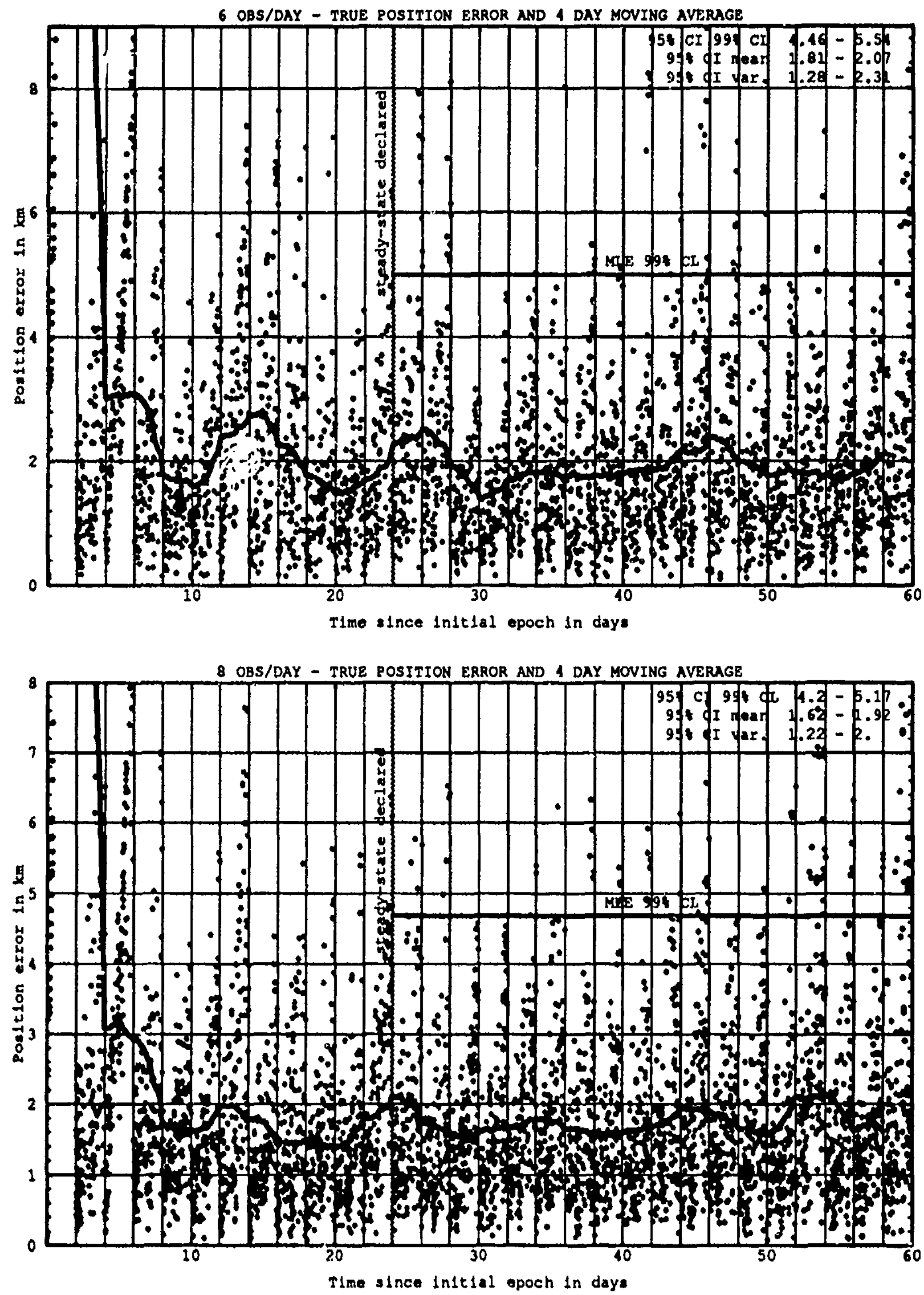


Figure H.65. Class: 3–1–2 (Catalog Number 14443), LUPI 2, OPD 6 and 8.

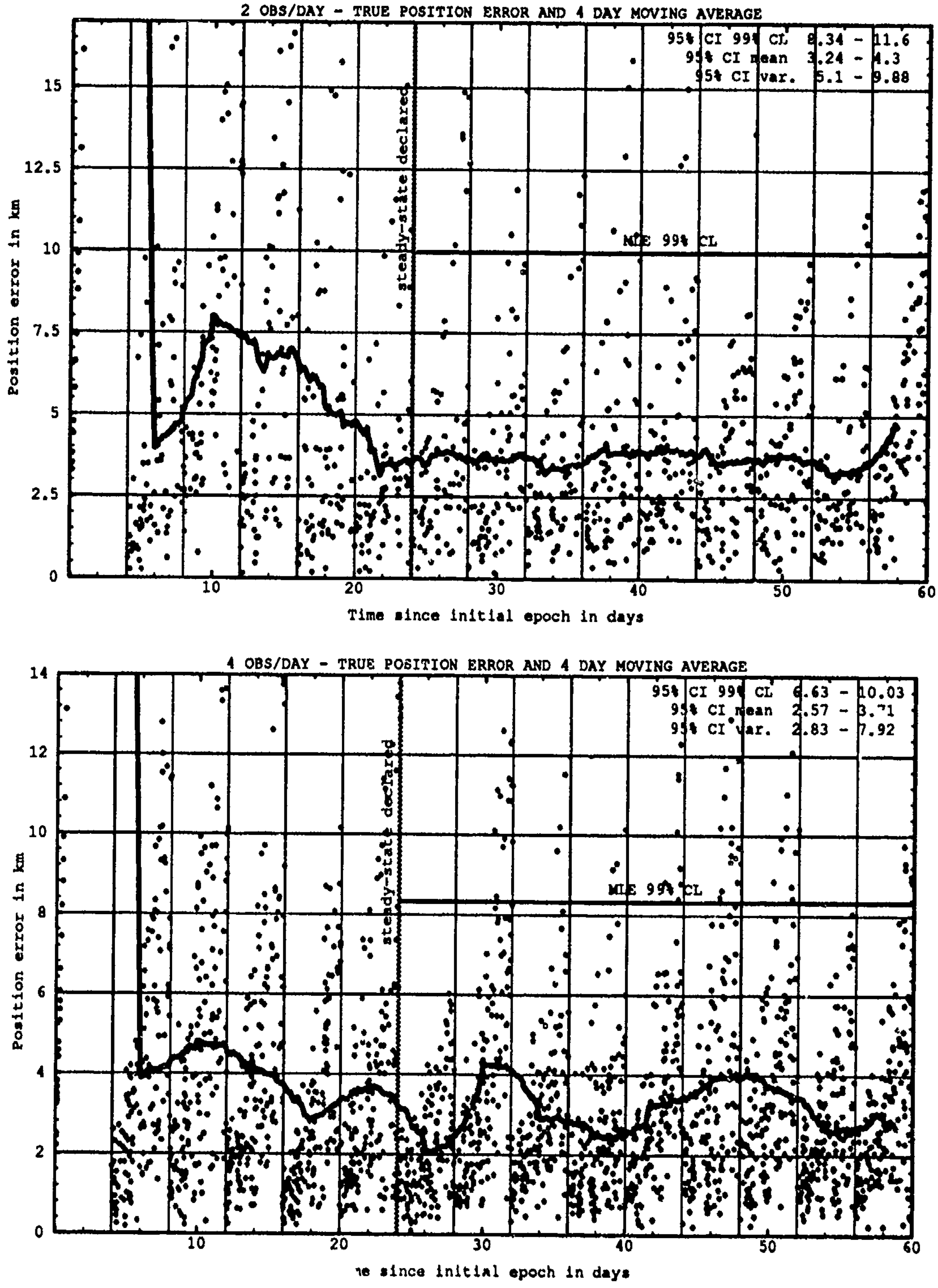


Figure H.66. Class: 3–1–2 (Catalog Number 14443), LUPI 4, OPD 2 and 4.

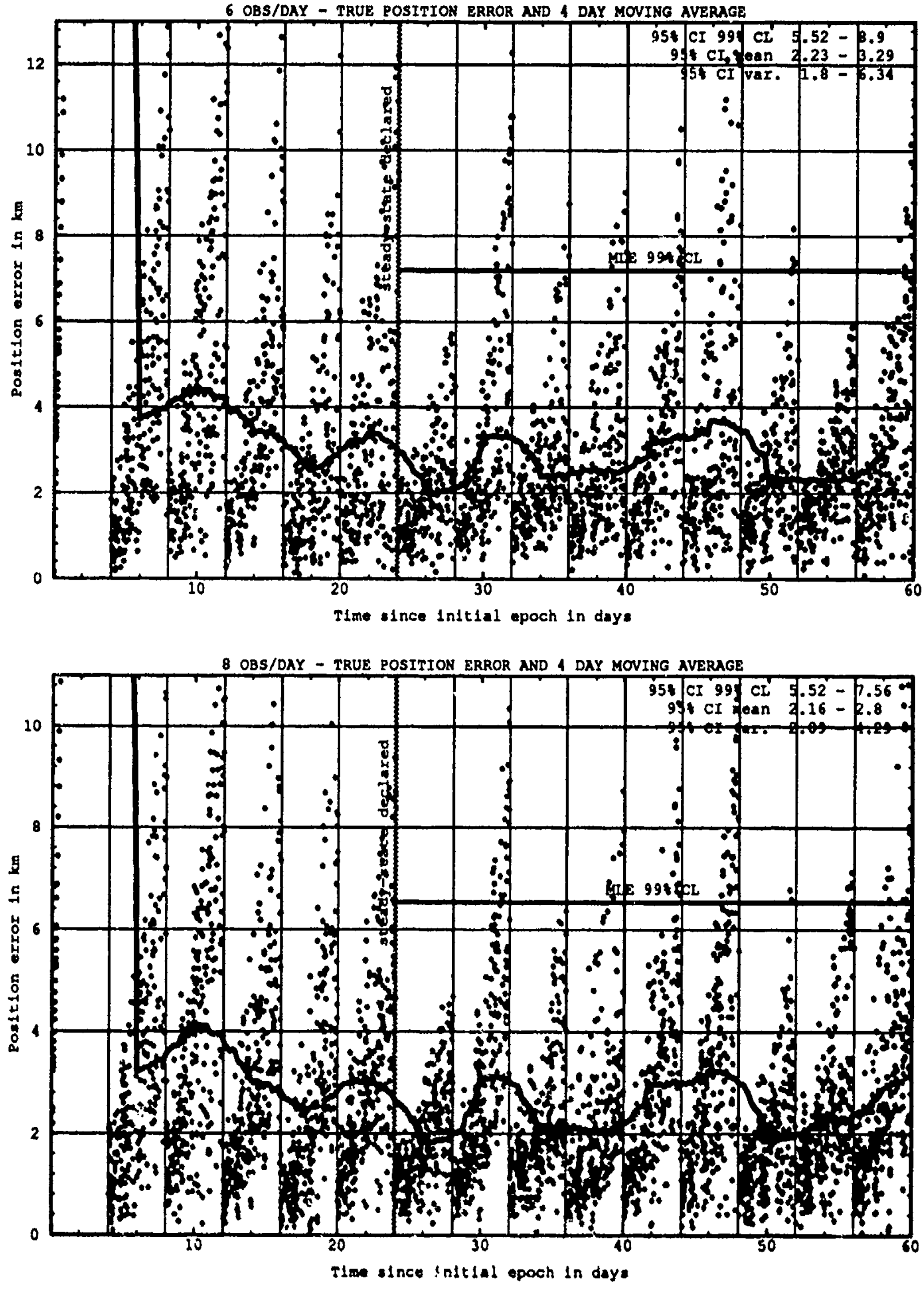
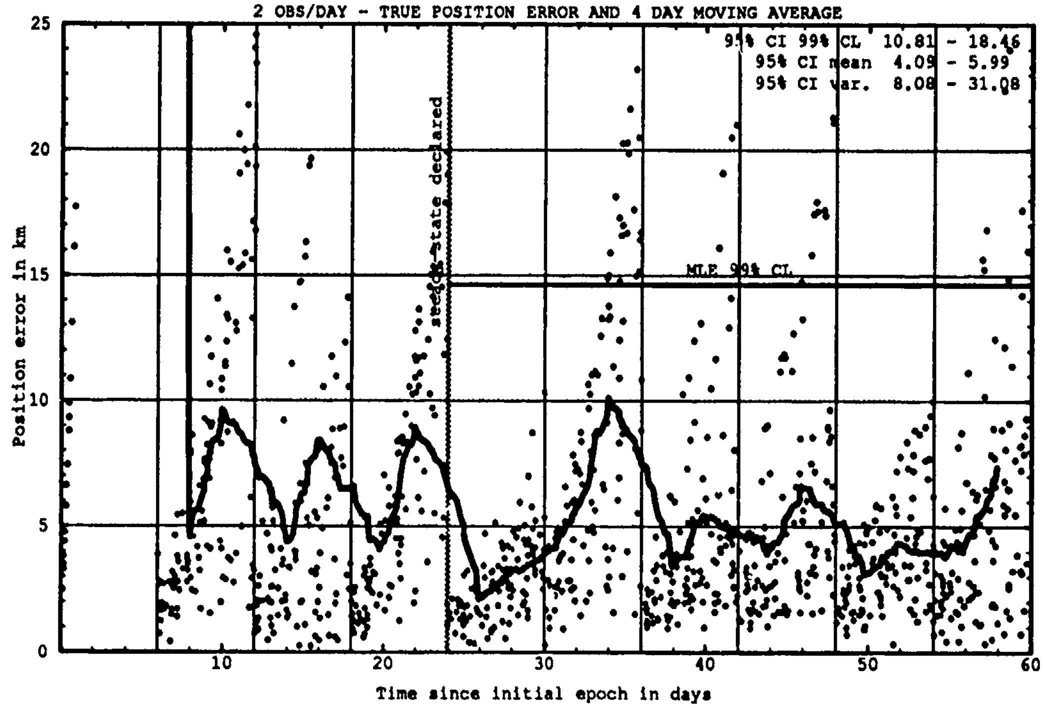


Figure H.67. Class: 3-1-2 (Catalog Number 14443), LUPI 4, OPD 6 and 8.



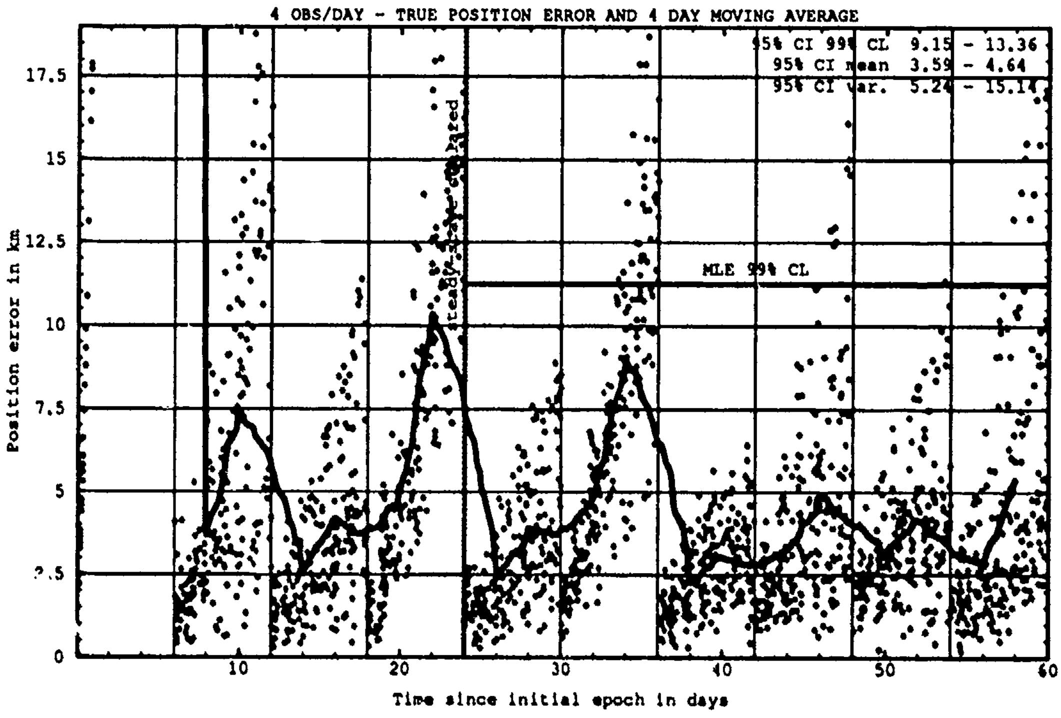
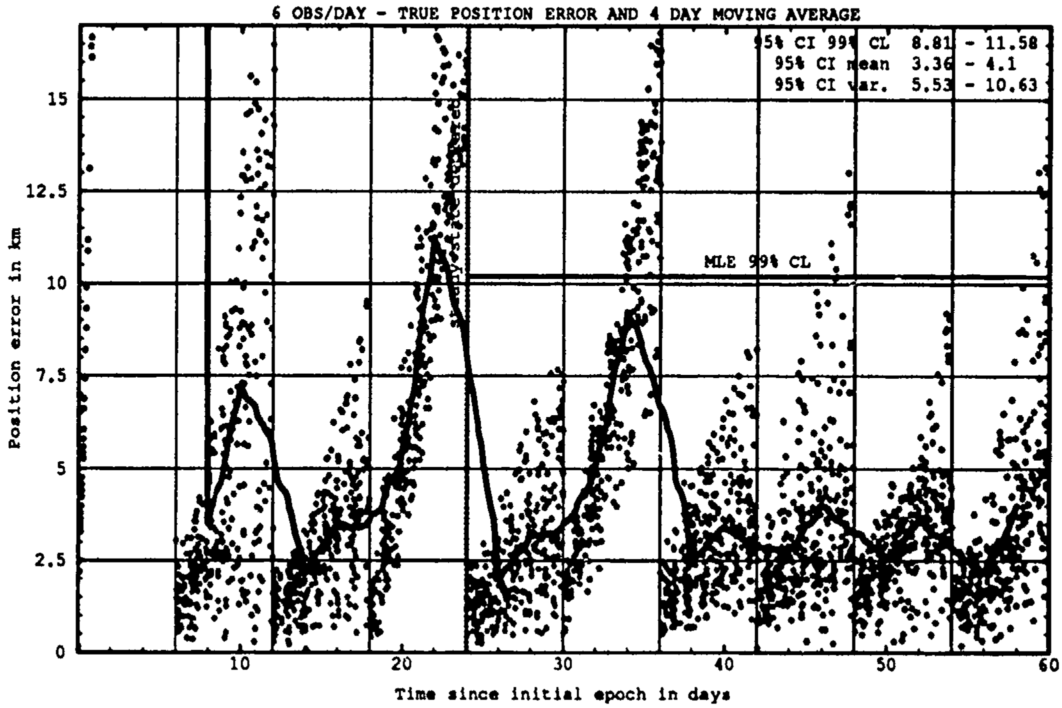


Figure H.68. Class: 3-1-2 (Catalog Number 14443), LUPI 6, OPD 2 and 4.



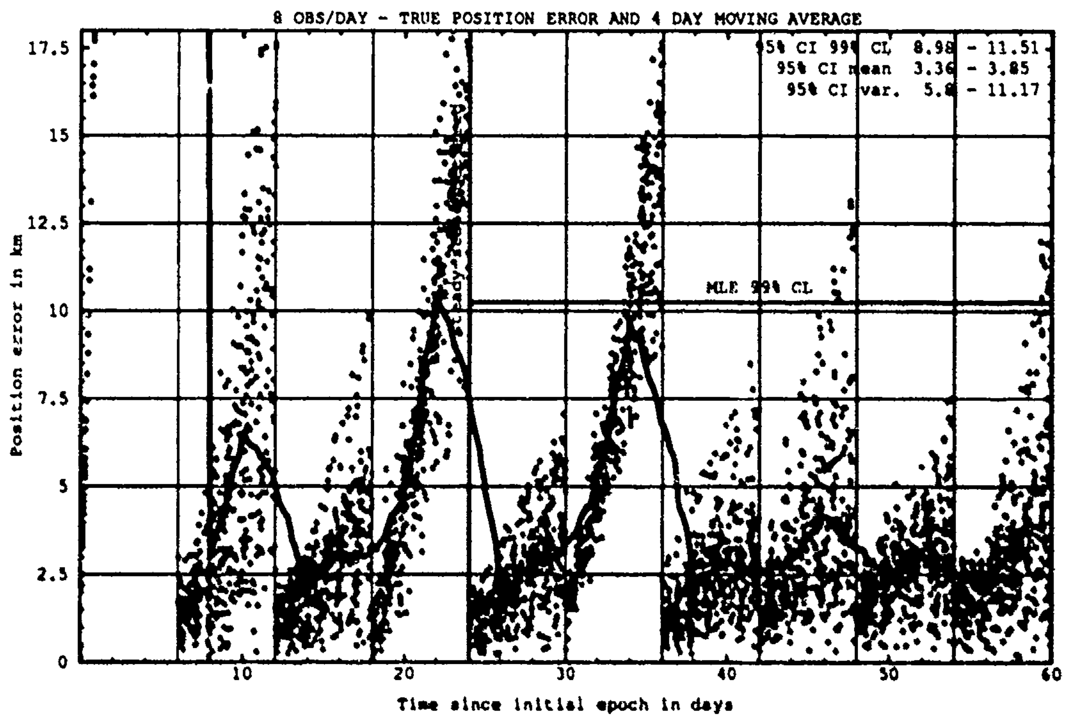


Figure H.69. Class: 3-1-2 (Catalog Number 14443), LUPI 6, OPD 6 and 8.

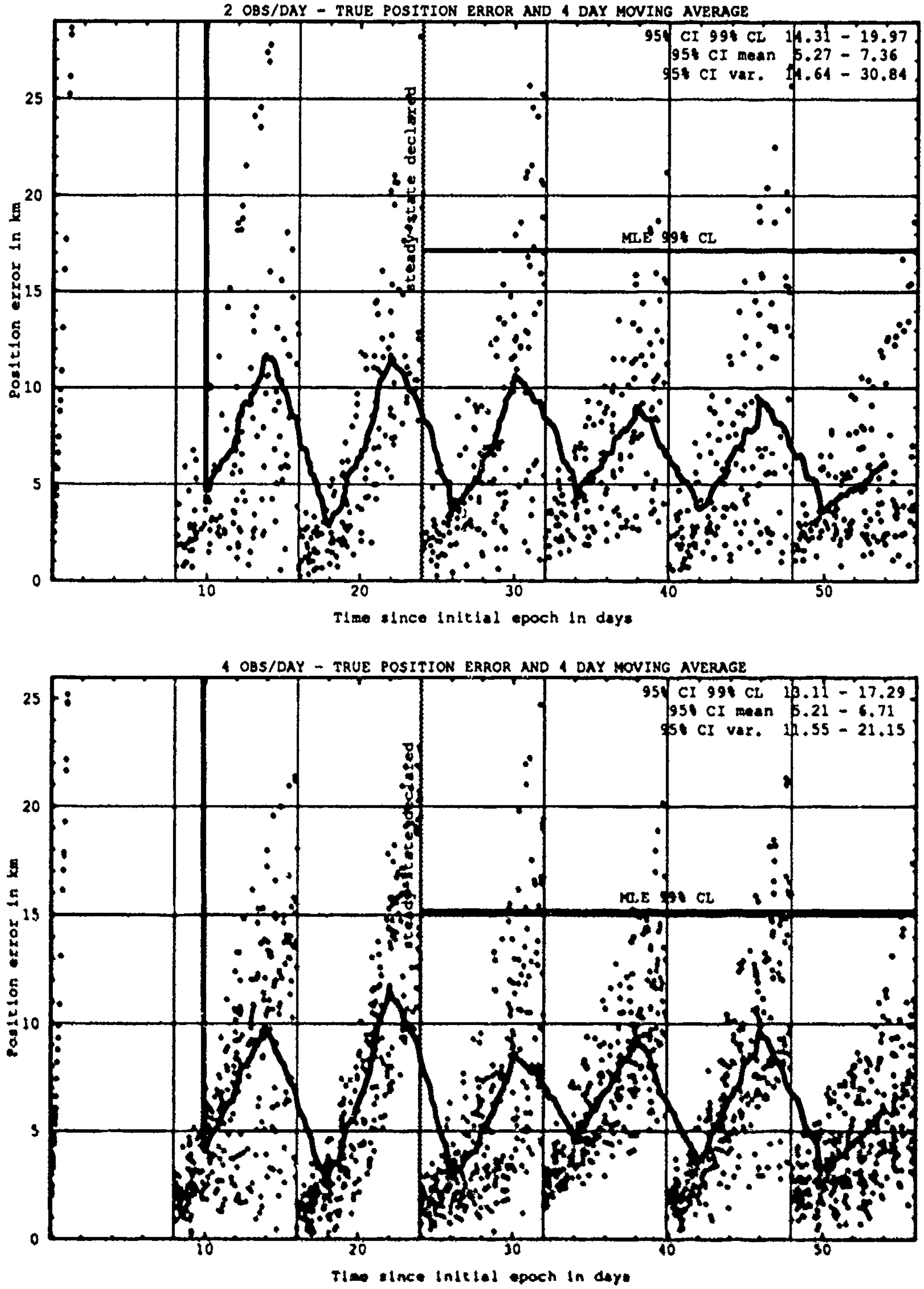


Figure H.70. Class: 3-1-2 (Catalog Number 14443), LUPI 8, OPD 2 and 4.

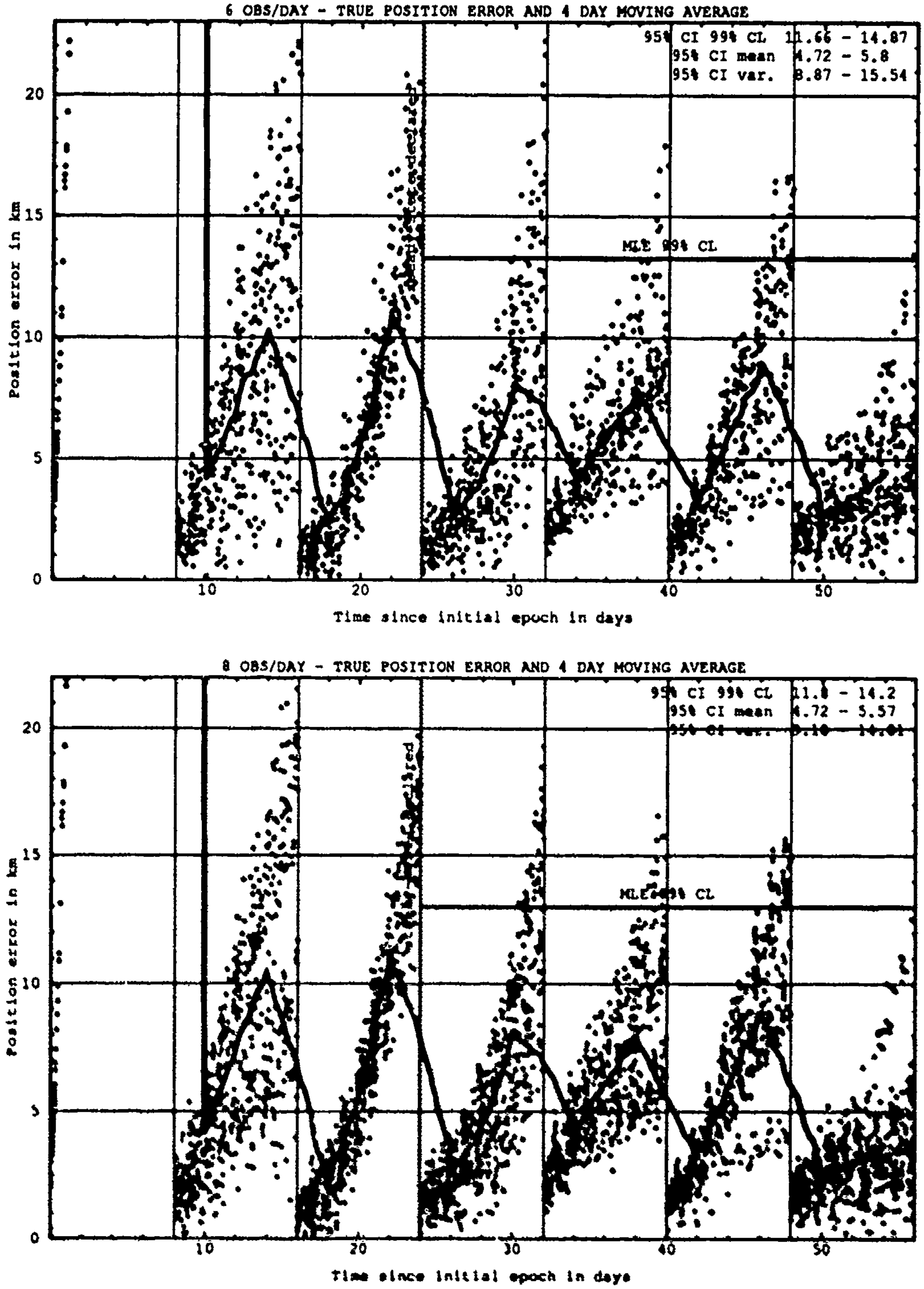


Figure H.71. Class: 3-1-2 (Catalog Number 14443), LUPI 8, OPD 6 and 8.

H.8.2 Differential Corrector Performance. The table below provides statistical information on the last-pass residuals of the differential corrector.

Table H.16. Selected Last-Pass 95% Confidence Interval Statistics.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
14443	8	8	1.00	1.03	0.25	0.29	2.17	2.27

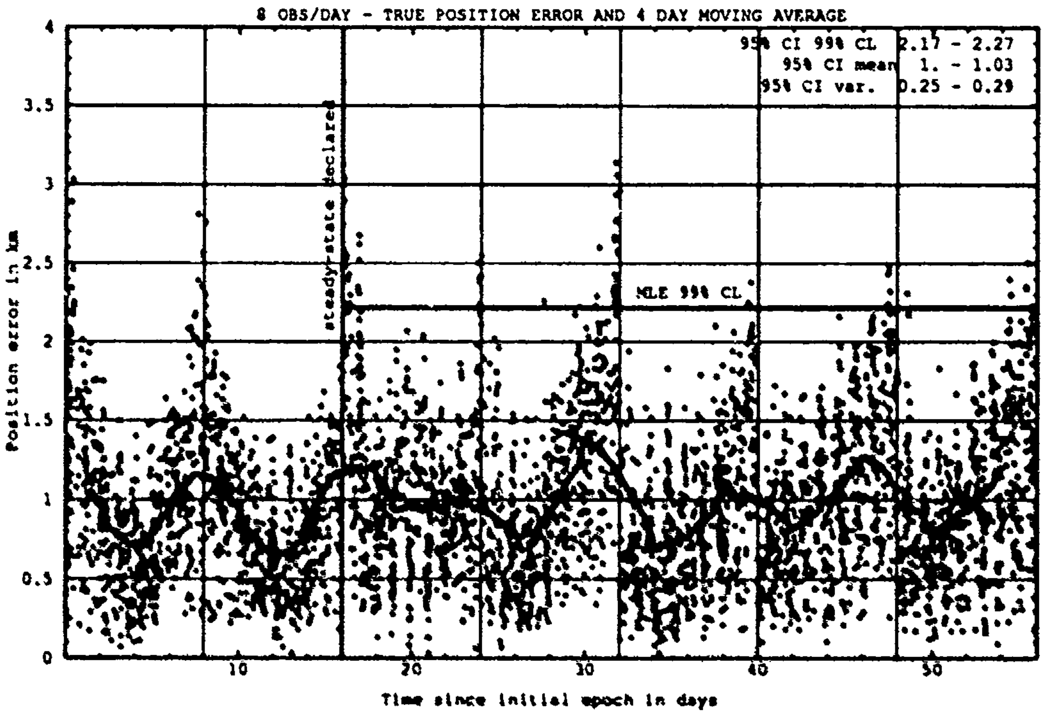


Figure H.72. Last-Pass — Class: 3-1-2 (Catalog Number 14443), LUPI 8, OPD 8.

H.9 CLASS: 3-1-3 (NORAD Catalog Number 19643)

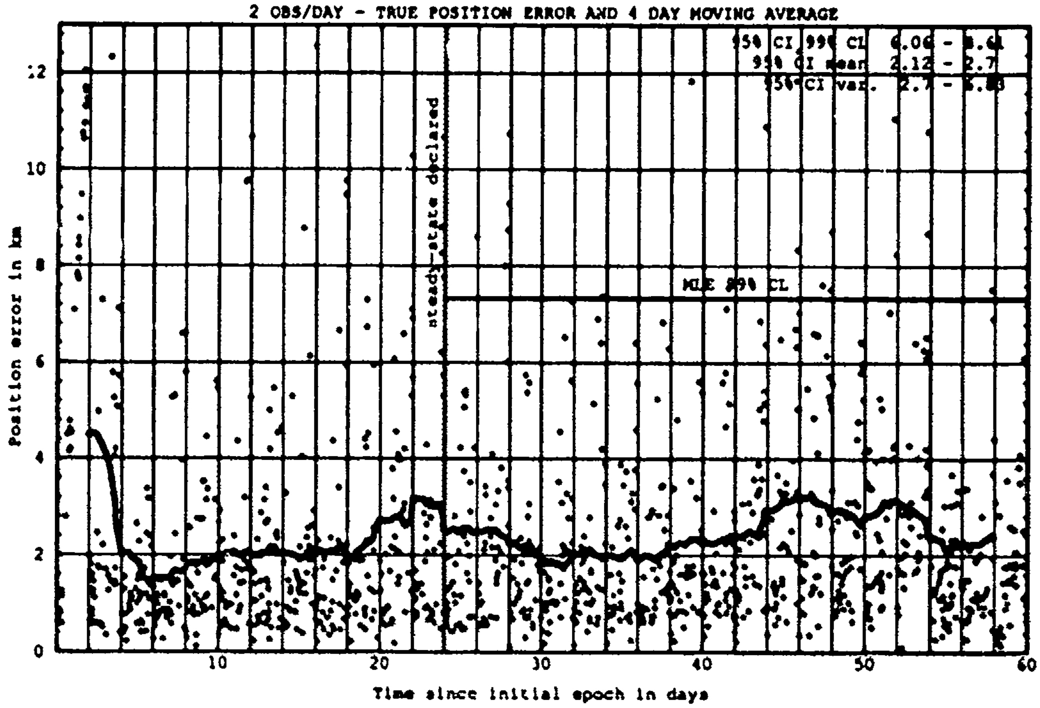
H.9.1 Prediction Performance. The tables below provide the statistical information on the ability of the estimates to predict for 1 LUPI after differential correction.

Table H.25. 95% Confidence Interval Analysis on Class: 3-1-3.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
19643	2	2	2.12	2.70	2.70	6.83	6.06	8.61
19643	2	4	1.41	1.93	1.24	2.33	4.01	5.43
19643	2	6	1.48	1.80	1.36	2.69	4.21	5.56
19643	2	8	1.38	1.64	1.35	2.06	4.08	4.95
19643	4	2	2.78	3.46	4.51	8.87	7.82	10.23
19643	4	4	2.27	2.85	3.52	6.69	6.72	8.74
19643	4	6	2.07	2.50	3.52	5.34	6.47	7.81
19643	4	8	1.81	2.24	2.51	4.27	5.52	6.98
19643	6	2	3.71	4.79	8.62	21.68	10.79	15.27
19643	6	4	3.49	4.22	7.64	14.64	10.03	12.94
19643	6	6	3.12	3.87	7.51	11.99	9.69	11.70
19643	6	8	3.13	3.68	6.93	10.65	9.33	11.15
19643	8	2	6.07	8.52	26.50	58.81	17.99	26.01
19643	8	4	5.18	6.93	18.81	33.05	15.25	20.15
19643	8	6	5.00	6.27	17.53	29.77	14.90	18.69
19643	8	8	5.05	5.94	17.79	29.31	14.98	18.34

Table H.26. ANOVA Analysis on Class: 3-1-3.

Source of Variation	Degrees of Freedom	Sum of Square	Mean Square	F	Table F
Replications	9	167.42	18.60	4.79	1.88
Main Effects:					
LUPI	3	3860.32	1286.77	331.02	2.60
OPD	3	279.89	93.29	24.00	2.60
Interaction	9	48.11	5.35	1.38	1.88
Error	135	524.78	3.89		
Total	159	4880.52			



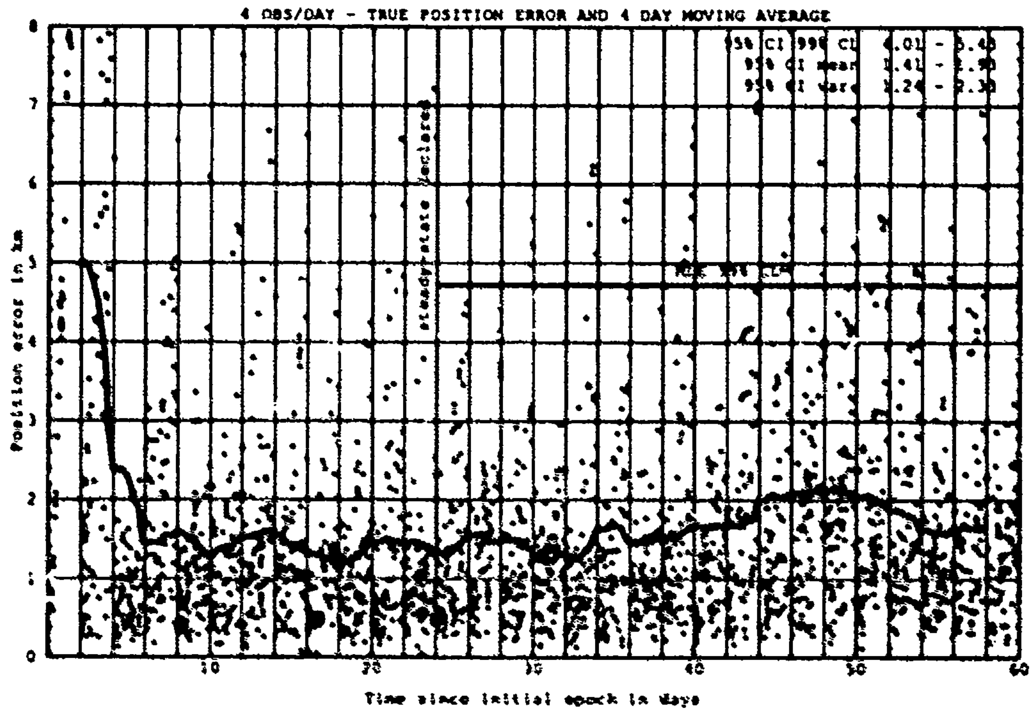


Figure II.73. Class: 3-1-3 (Catalog Number 19643), LUPI 2, OPD 2 and 4.

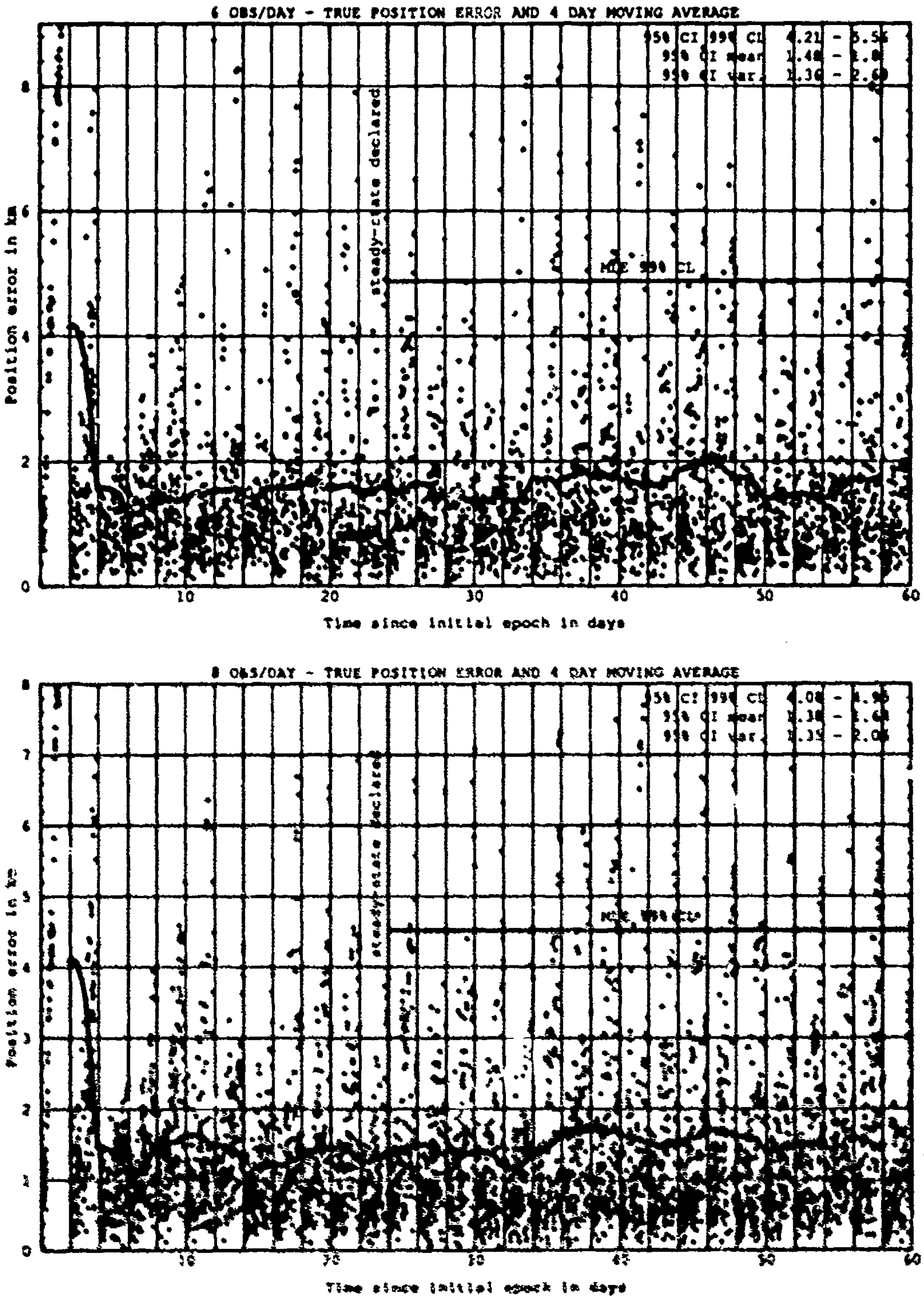
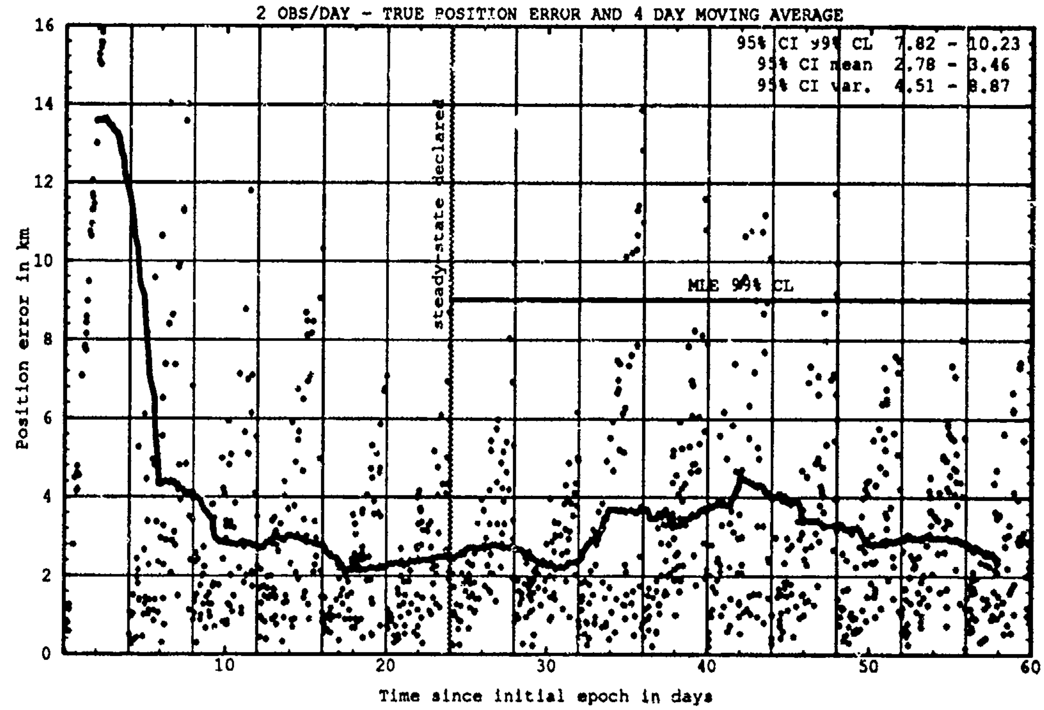


Figure H.74. Class: 3-1-3 (Catalog Number 19643), LUPI 2, OPD 6 and 8.



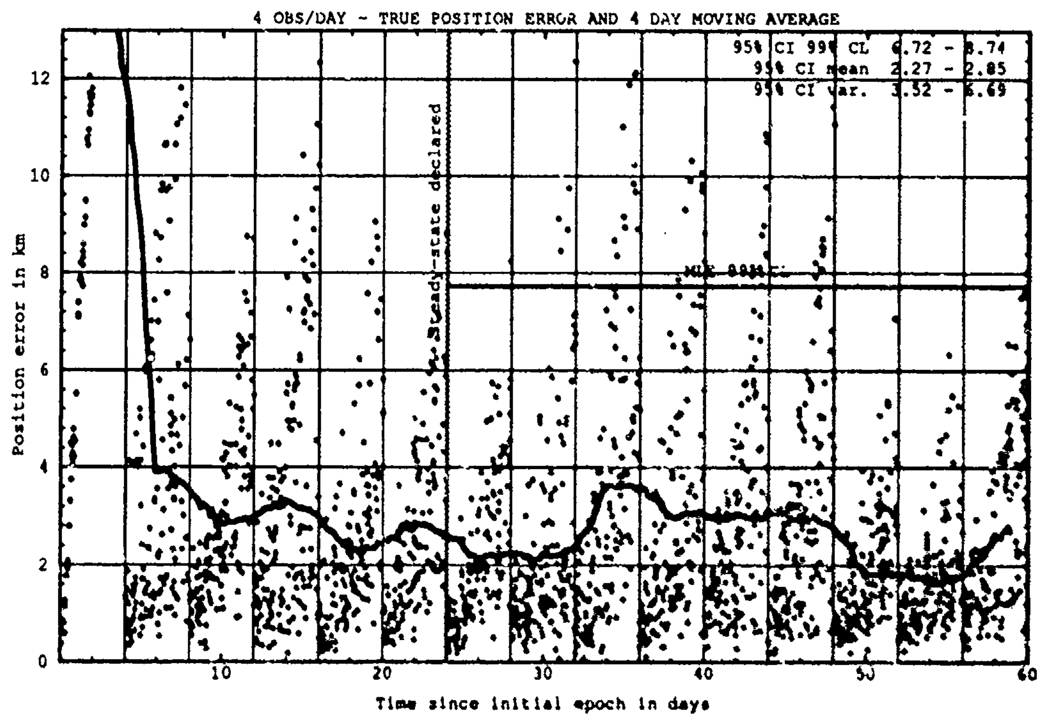
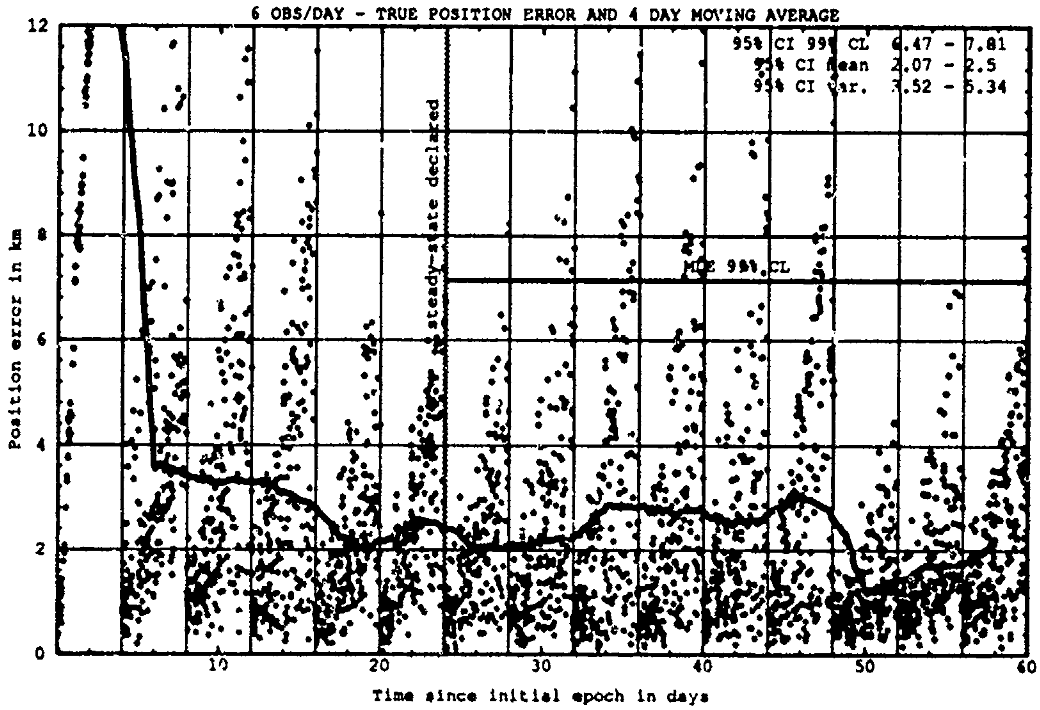


Figure H.75. Class: 3-1-3 (Catalog Number 19643), LUPI 4, OPD 2 and 4.



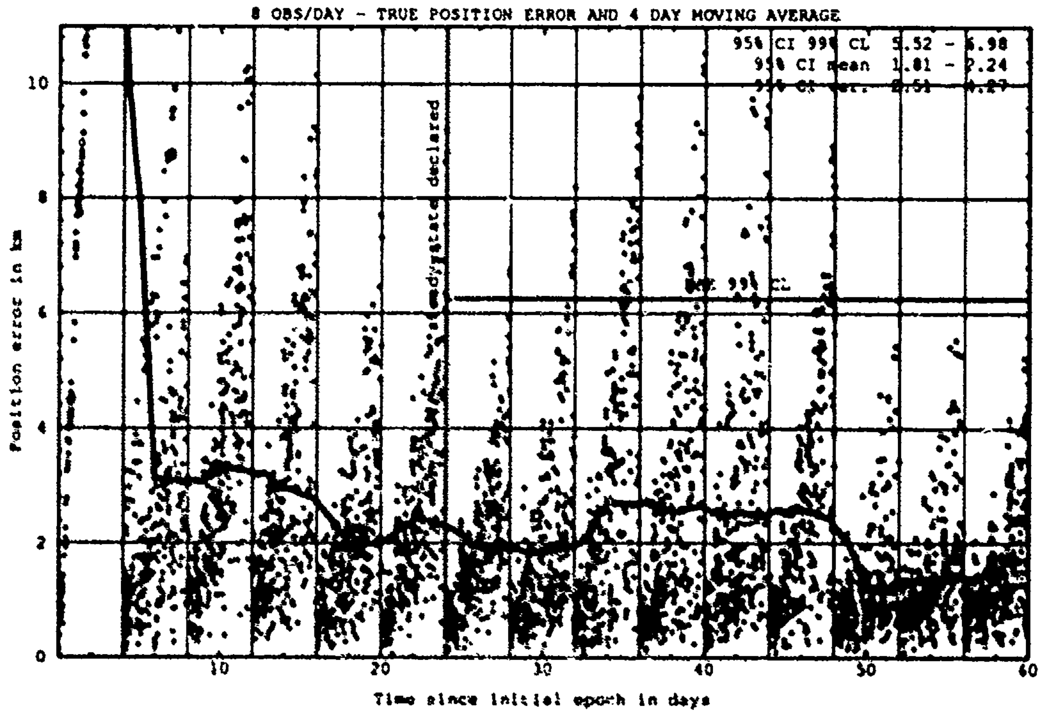


Figure H.76. Class: 3-1-3 (Catalog Number 19643), LUPI 4, OPD 6 and 8.

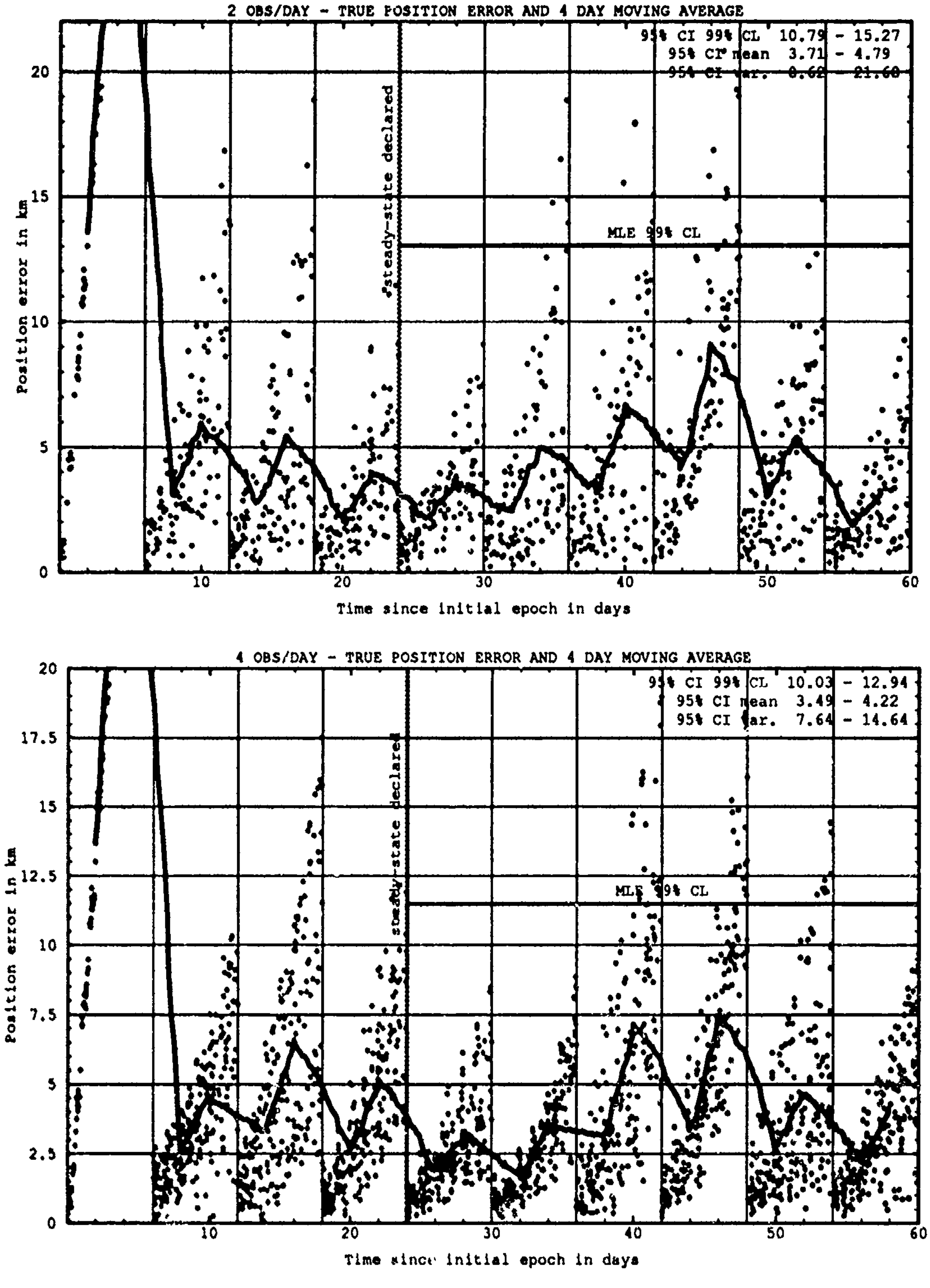
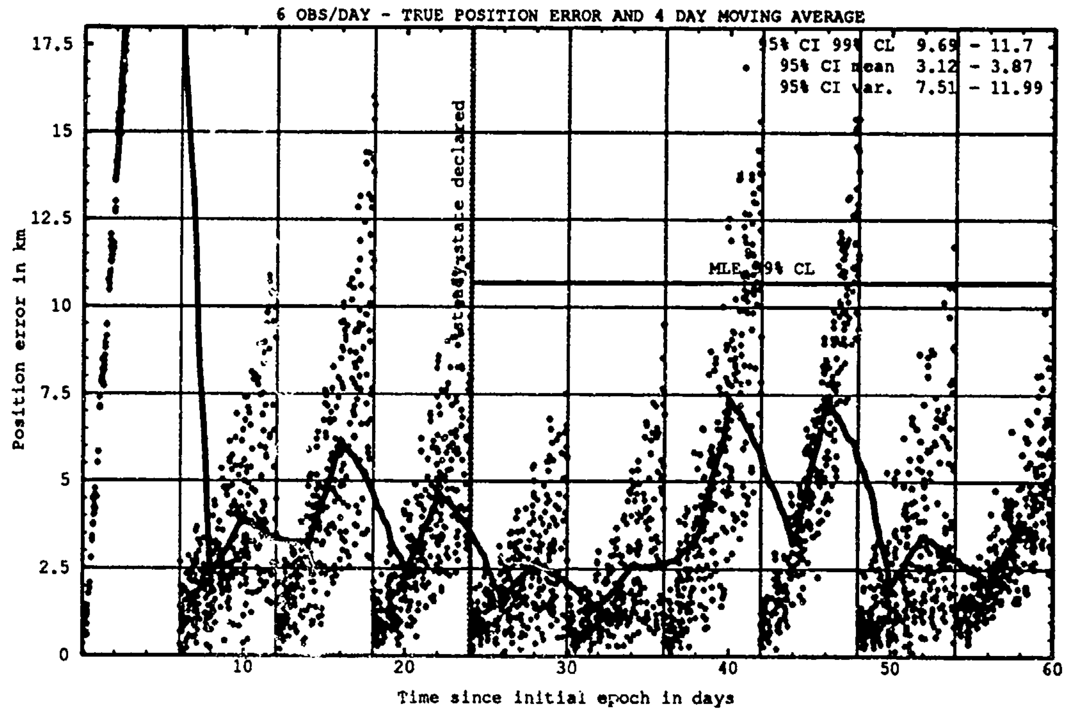


Figure H.77. Class: 3–1–3 (Catalog Number 19643), LUPI 6, OPD 2 and 4.



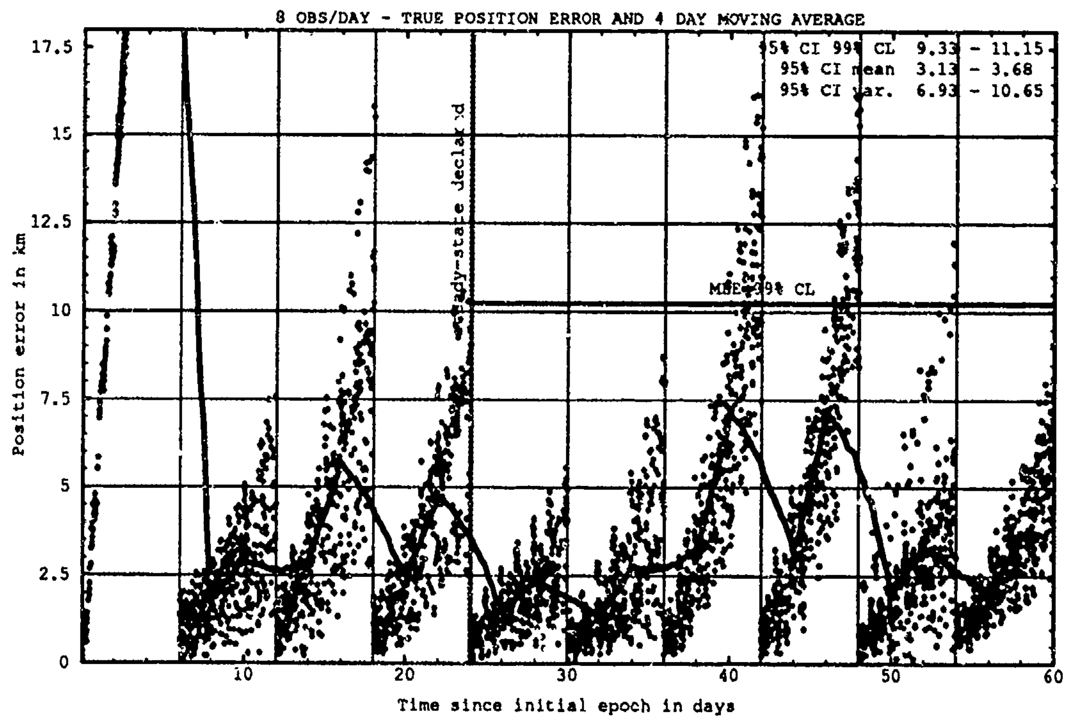
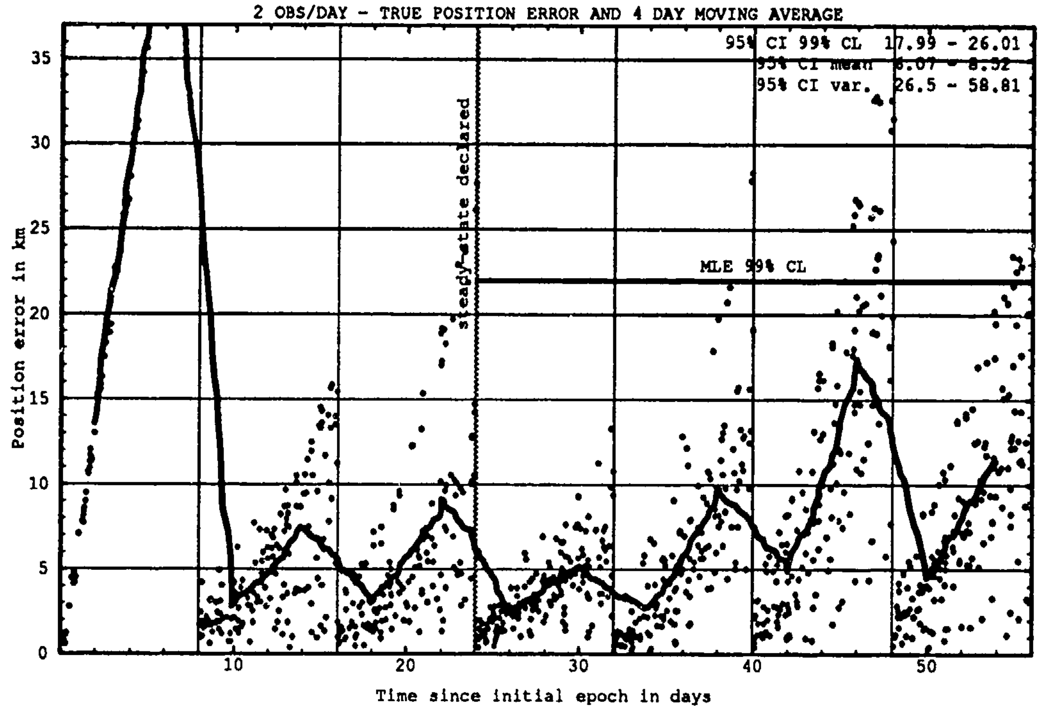


Figure H.78. Class: 3-1-3 (Catalog Number 19643), LUPI 6, OPD 6 and 8.



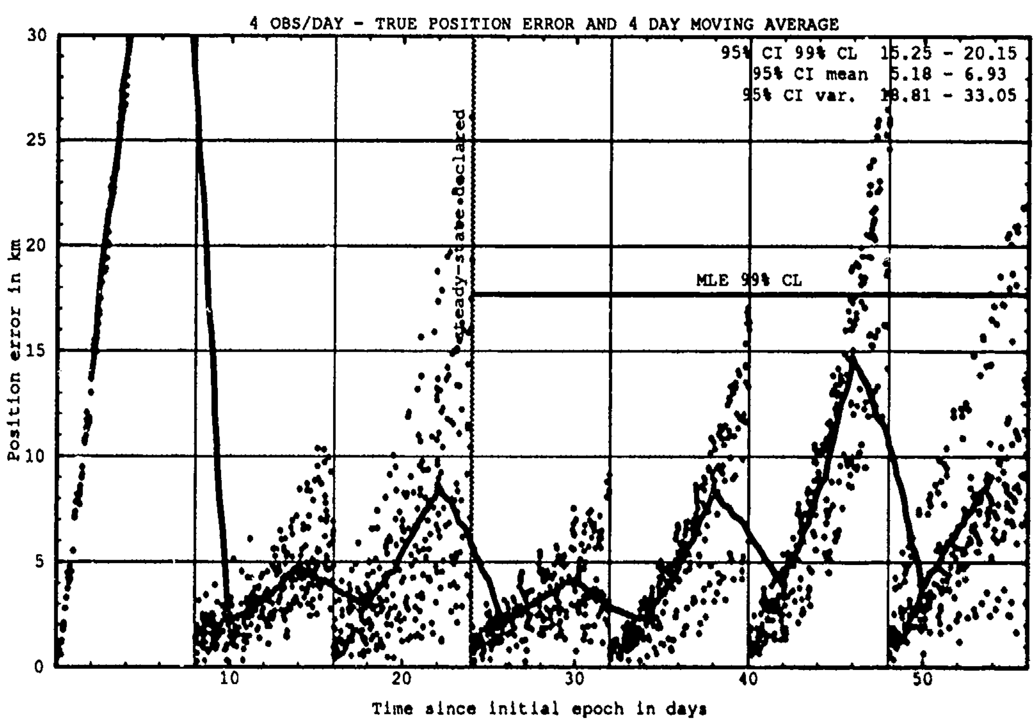


Figure H.79. Class: 3–1–3 (Catalog Number 19643), LUPI 8, OPD 2 and 4.

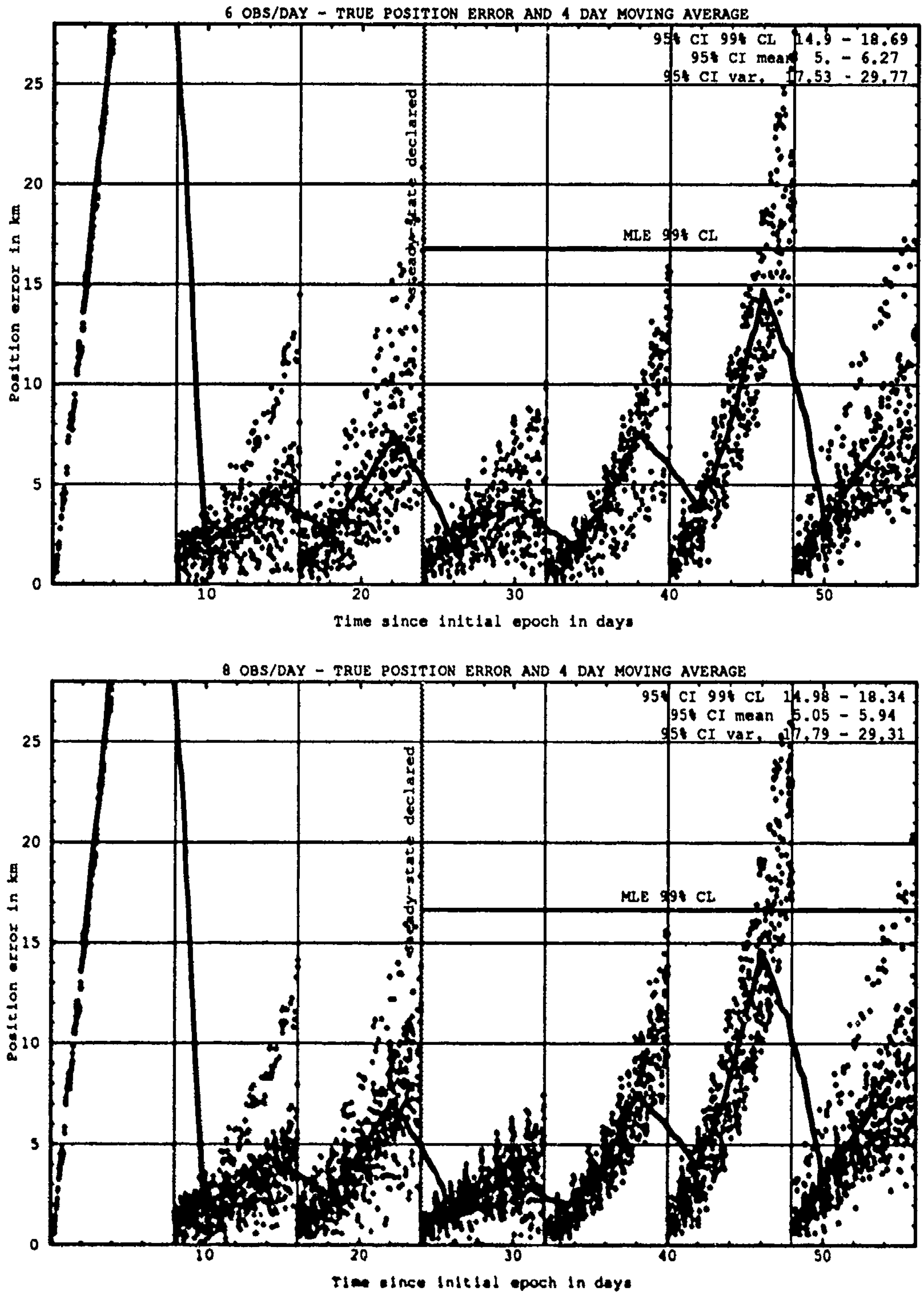


Figure II.80. Class: 3-1-3 (Catalog Number 19643), LUPI 8, OPD 6 and 8.

H.9.2 Differential Corrector Performance. The table below provides statistical information on the last-pass residuals of the differential corrector.

Table H.18. Selected Last-Pass 95% Confidence Interval Statistics.

Catalog Number	LUPI	OPD	Mean Min	Mean Max	Variance Min	Variance Max	99% CL Min	99% CL Max
	days	obs/day	km	km	km^2	km^2	km	km
19643	8	8	0.73	0.76	0.13	0.15	1.58	1.66

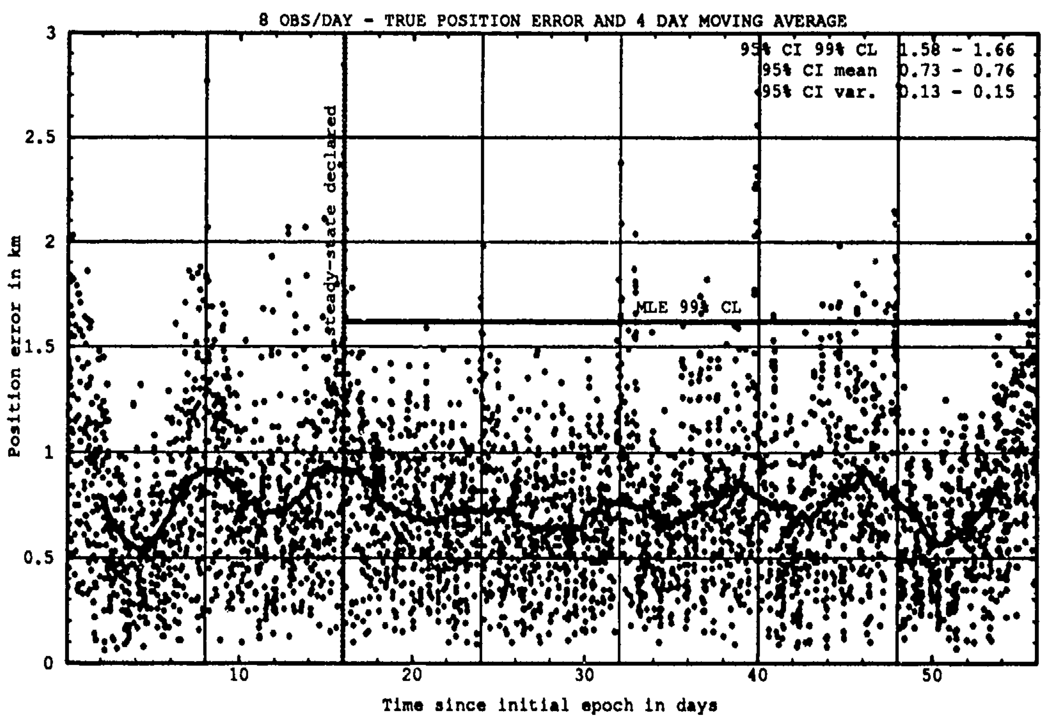


Figure H.81. Last-Pass — Class: 3–1–3 (Catalog Number 19643), LUPI 8, OPD 8.

H.10 CLASS: 3-1-4 (NORAD Catalog Number 17429)

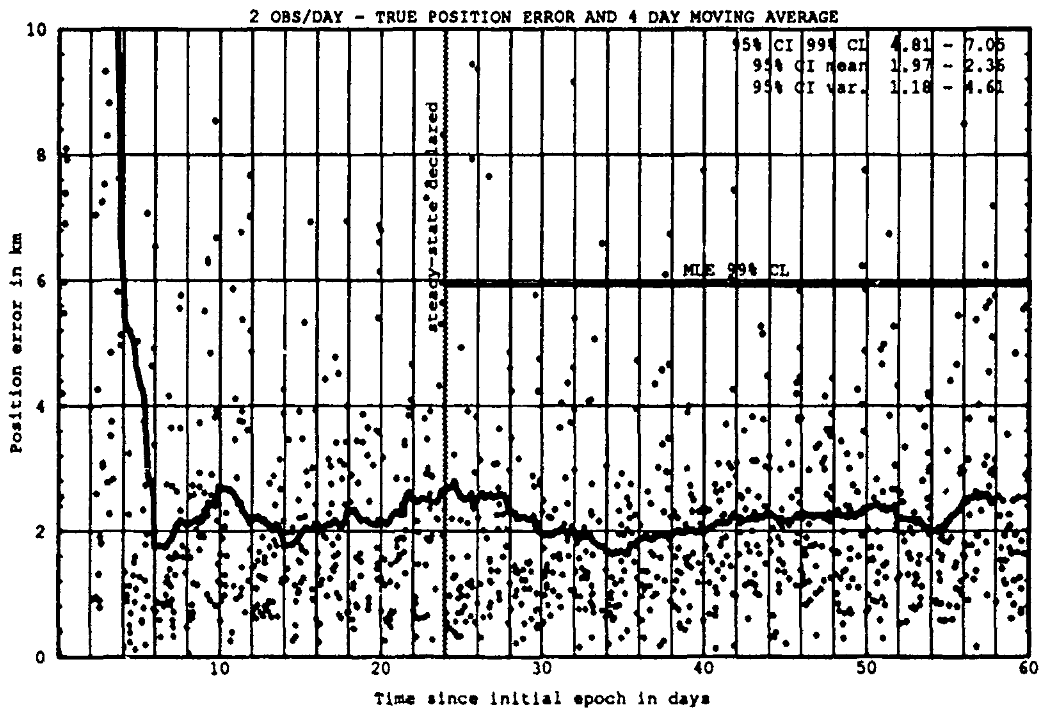
H.10.1 Prediction Performance. The tables below provide the statistical information on the ability of the estimates to predict for 1 LUPI after differential correction.

Table H.28. 95% Confidence Interval Analysis on Class: 3-1-4.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
17429	2	2	1.97	2.36	1.18	4.61	4.81	7.05
17429	2	4	1.73	2.04	0.96	3.24	4.25	6.01
17429	2	6	1.64	2.08	1.15	3.02	4.23	5.98
17429	2	8	1.73	1.94	1.59	2.36	4.68	5.49
17429	4	2	3.03	4.17	4.56	20.02	8.39	14.05
17429	4	4	3.17	3.85	6.58	10.25	9.13	11.25
17429	4	6	2.78	3.52	4.65	9.50	7.92	10.51
17429	4	8	2.64	3.16	4.00	6.95	7.36	9.18
17429	6	2	4.33	5.90	9.91	30.32	12.13	18.15
17429	6	4	4.16	5.23	10.30	27.97	12.12	17.02
17429	6	6	3.95	4.78	10.40	21.46	11.63	15.29
17429	6	8	3.71	4.38	8.47	20.53	10.87	14.52
17429	8	2	5.64	8.82	16.94	59.17	15.77	25.93
17429	8	4	5.35	6.58	21.03	30.86	16.22	19.22
17429	8	6	5.32	6.15	18.44	30.12	15.45	18.67
17429	8	8	4.79	5.93	18.65	26.07	14.97	17.62

Table H.29. ANOVA Analysis on Class: 3-1-4.

Source of Variation	Degrees of Freedom	Sum of Square	Mean Square	F	Table F
Replications	9	266.84	29.65	4.88	1.88
Main Effects:					
LUPI	3	3573.71	1191.24	195.95	2.60
OPD	3	160.89	53.63	8.82	2.60
Interaction	9	48.39	5.34	0.8844	1.88
Error	135	820.71	6.08		
Total	159	4870.53			



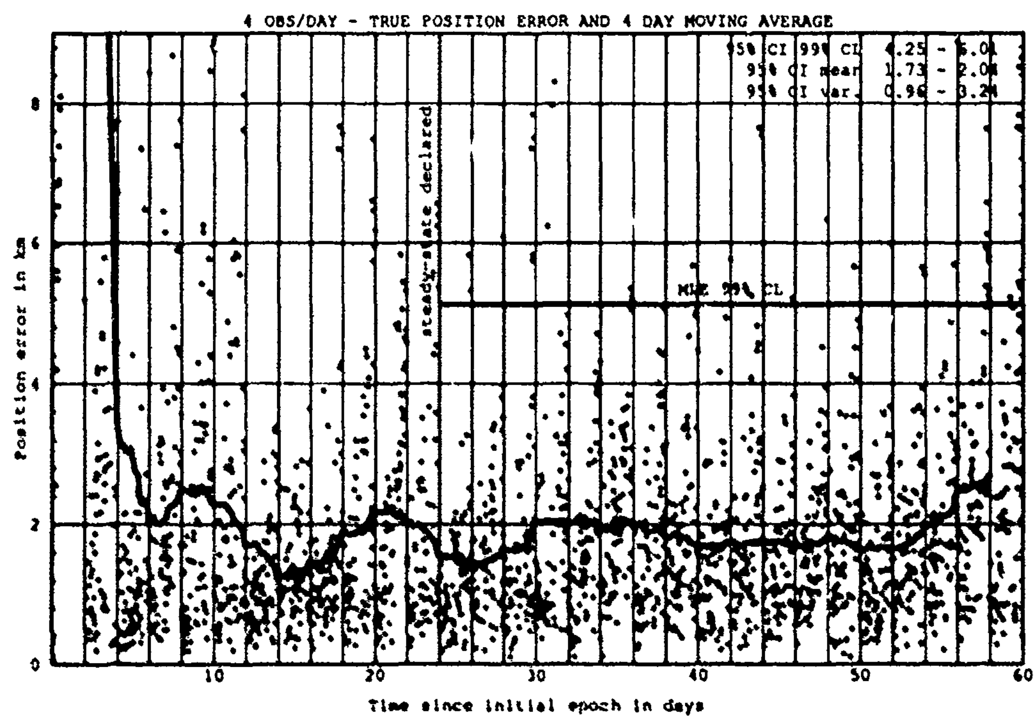
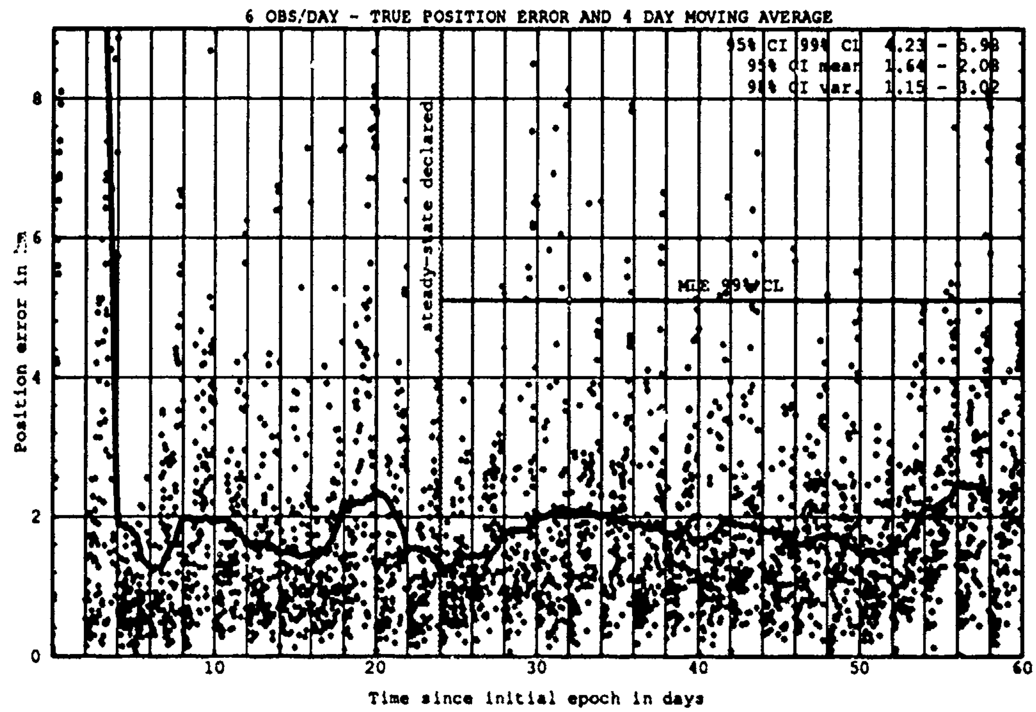


Figure H.82. Class: 3-1-4 (Catalog Number 17429), LUPI 2, OPD 2 and 4.



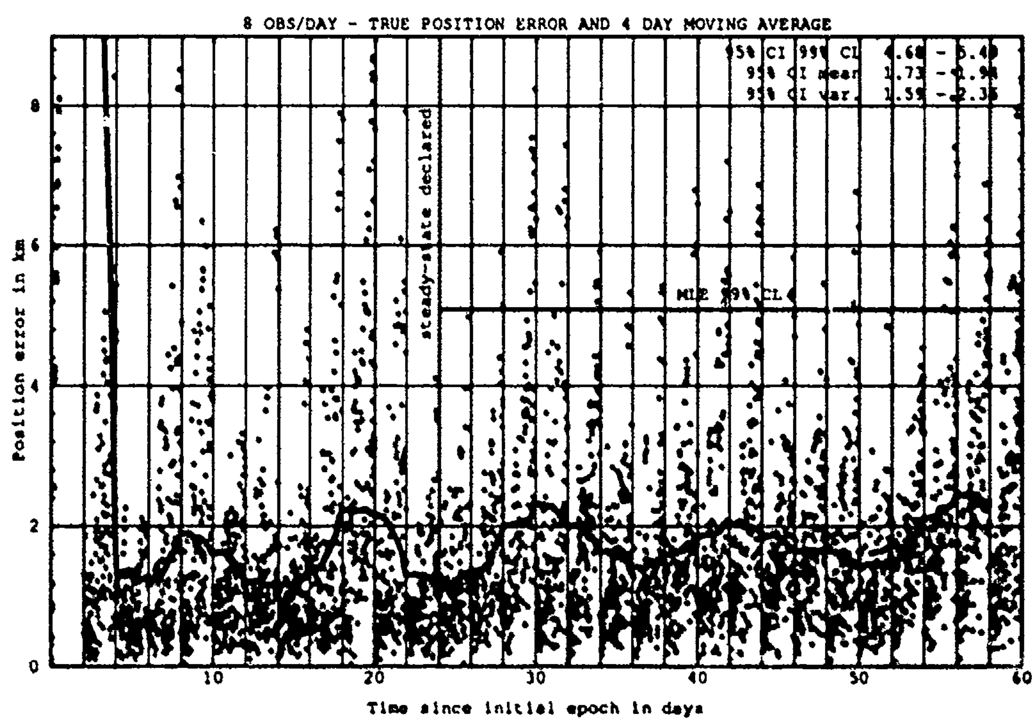


Figure II.83. Class: 3-1-4 (Catalog Number 17429), LUPI 2, OPD 6 and 8.

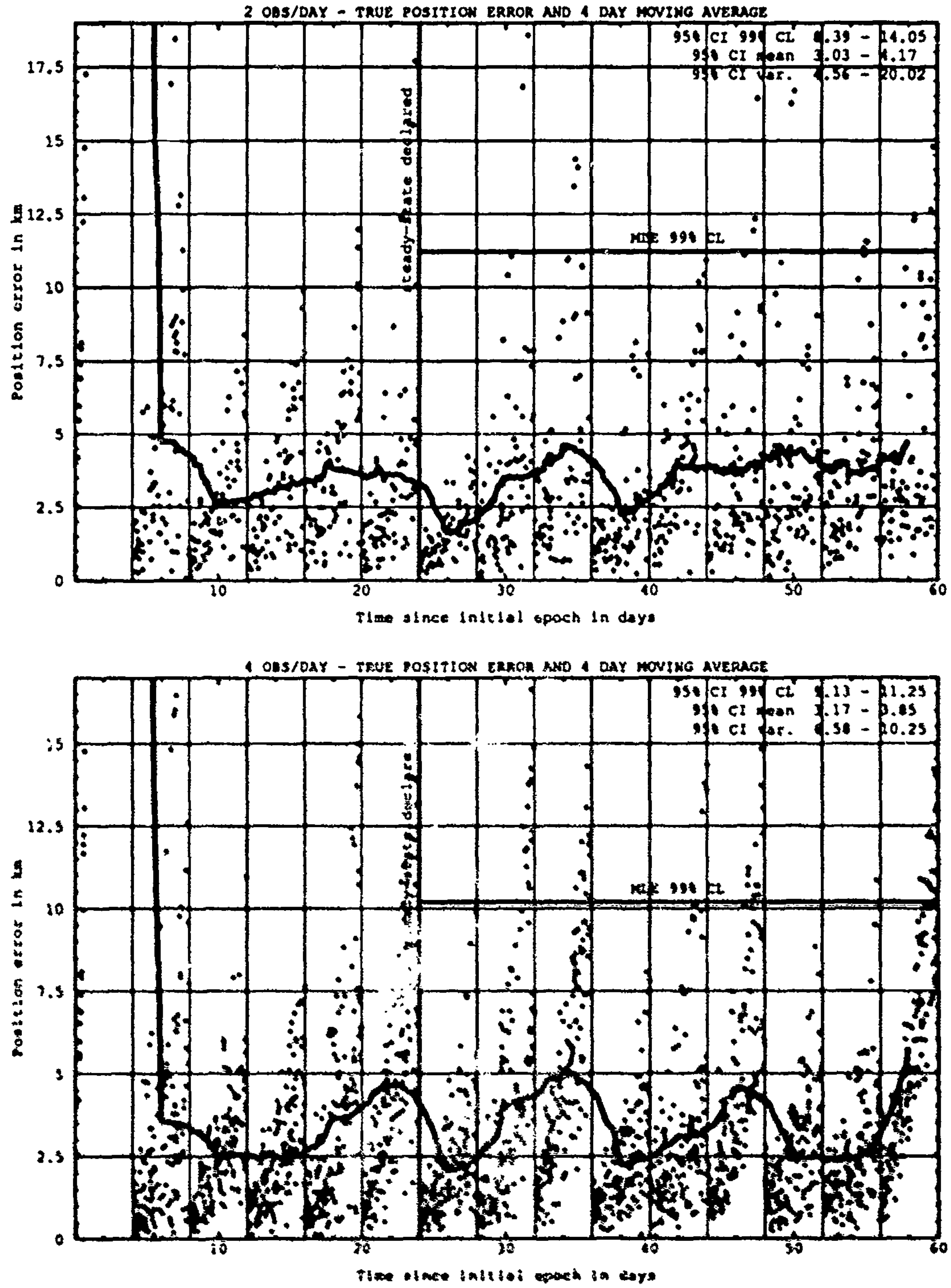
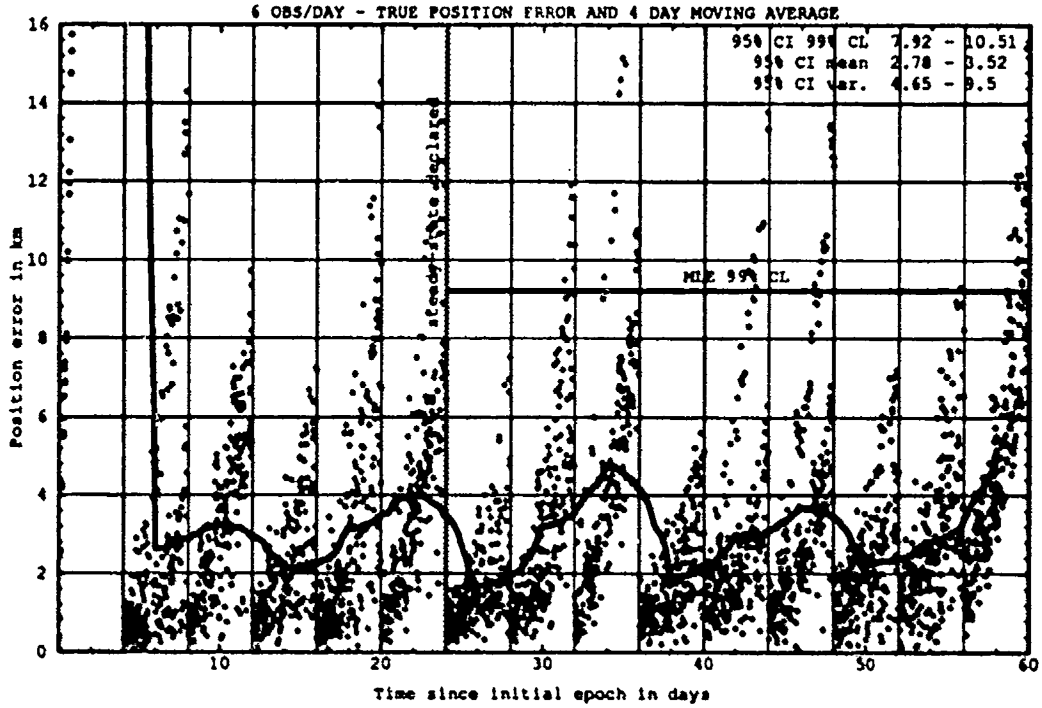


Figure H.84. Class: 3-1-4 (Catalog Number 17429), LUPI 4, OPD 2 and 4.



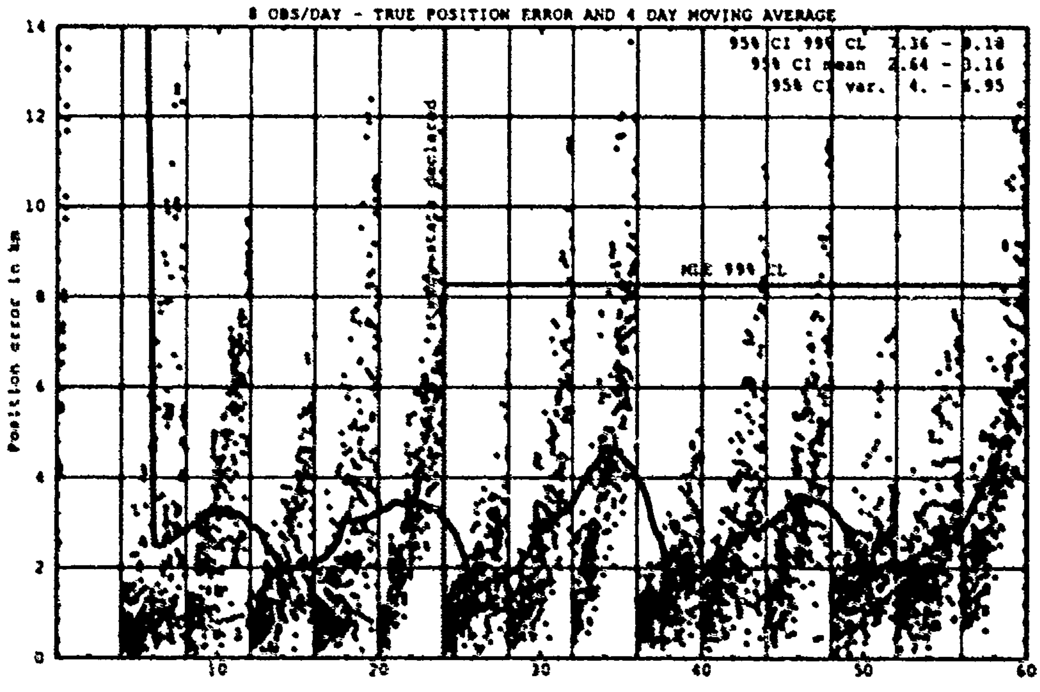


Figure H.85. Class: 3-1-4 (Catalog Number 17429), LUPI 4, OPD 6 and 8.

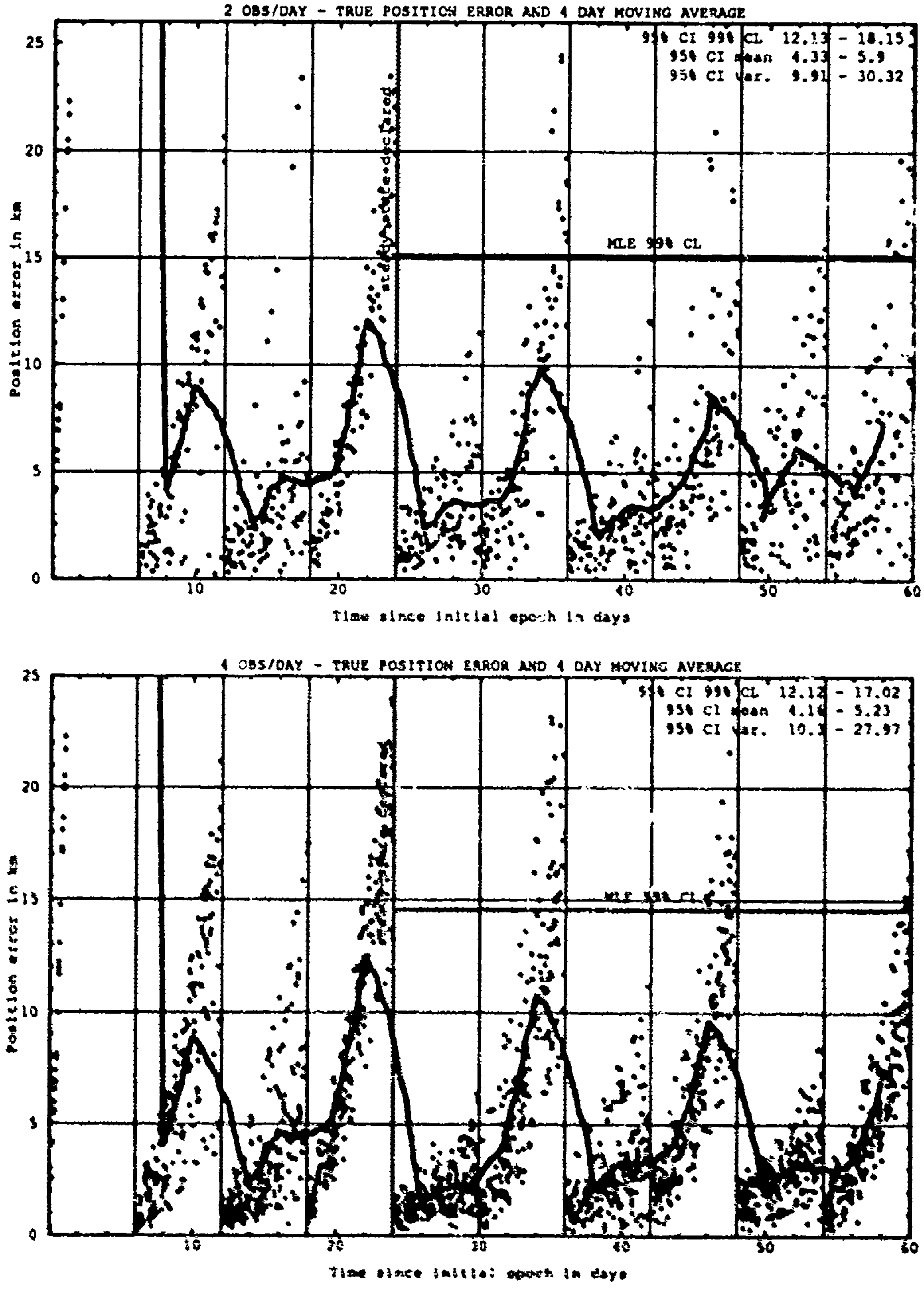


Figure H.86. Class: 3-1-4 (Catalog Number 17429), LUPI 6, OPD 2 and 4.

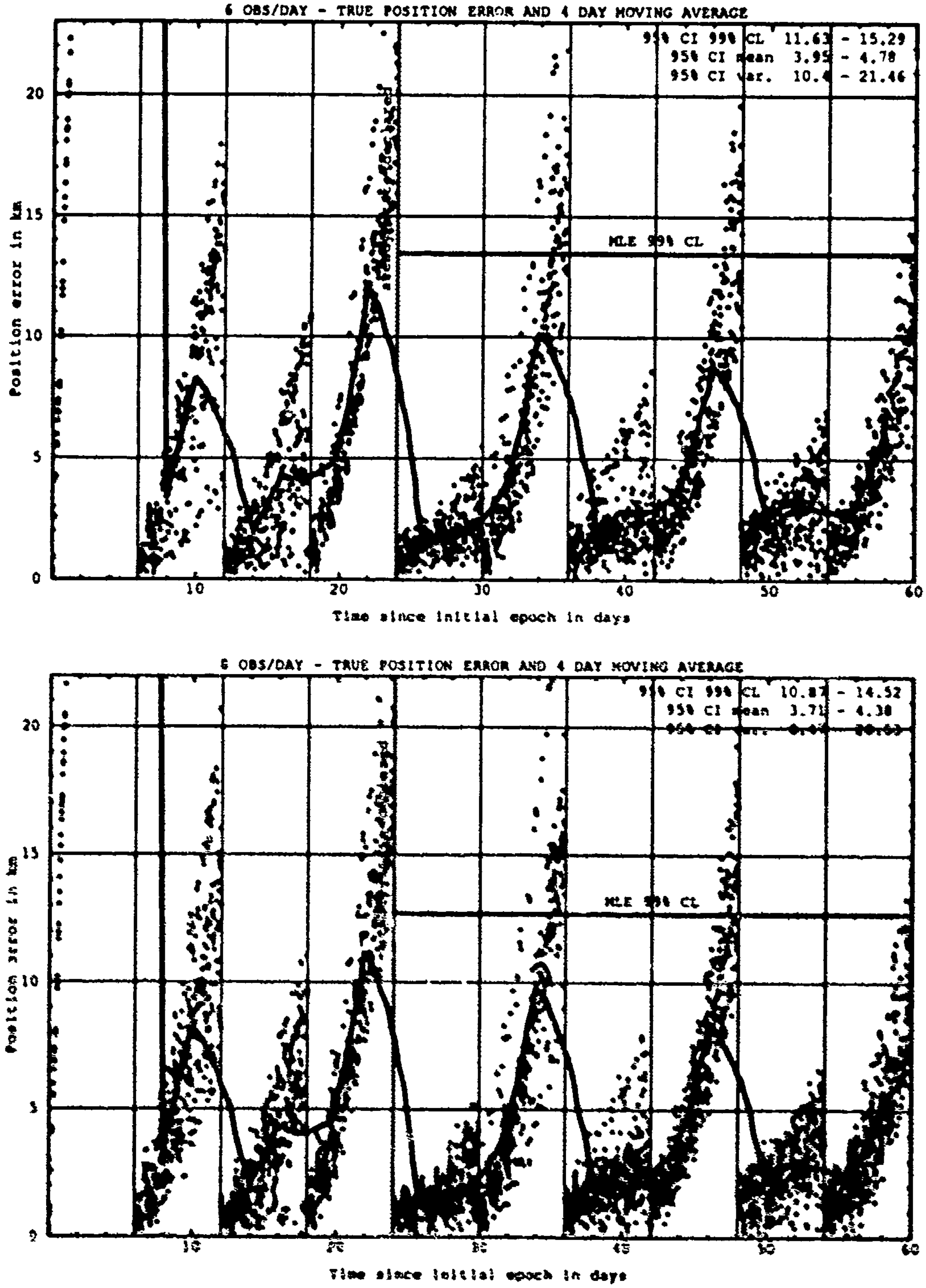


Figure H.87 Class: 3-1-4 (Catalog Number 17429), LUPI 6, OPD 6 and 8.

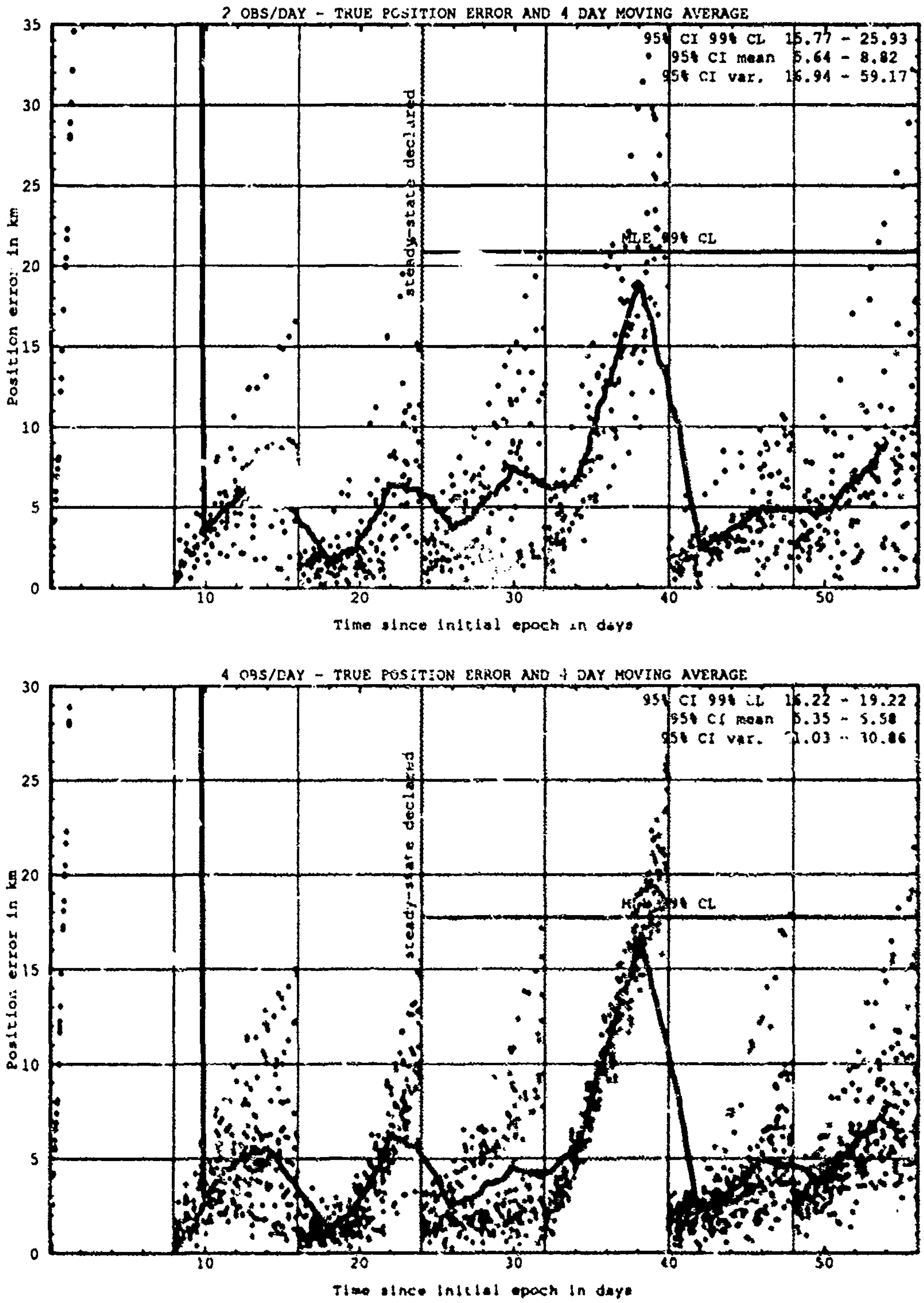


Figure II.88. Class: 3-1-4 (Catalog Number 17429), LUPI 8, OPD 2 and 4.

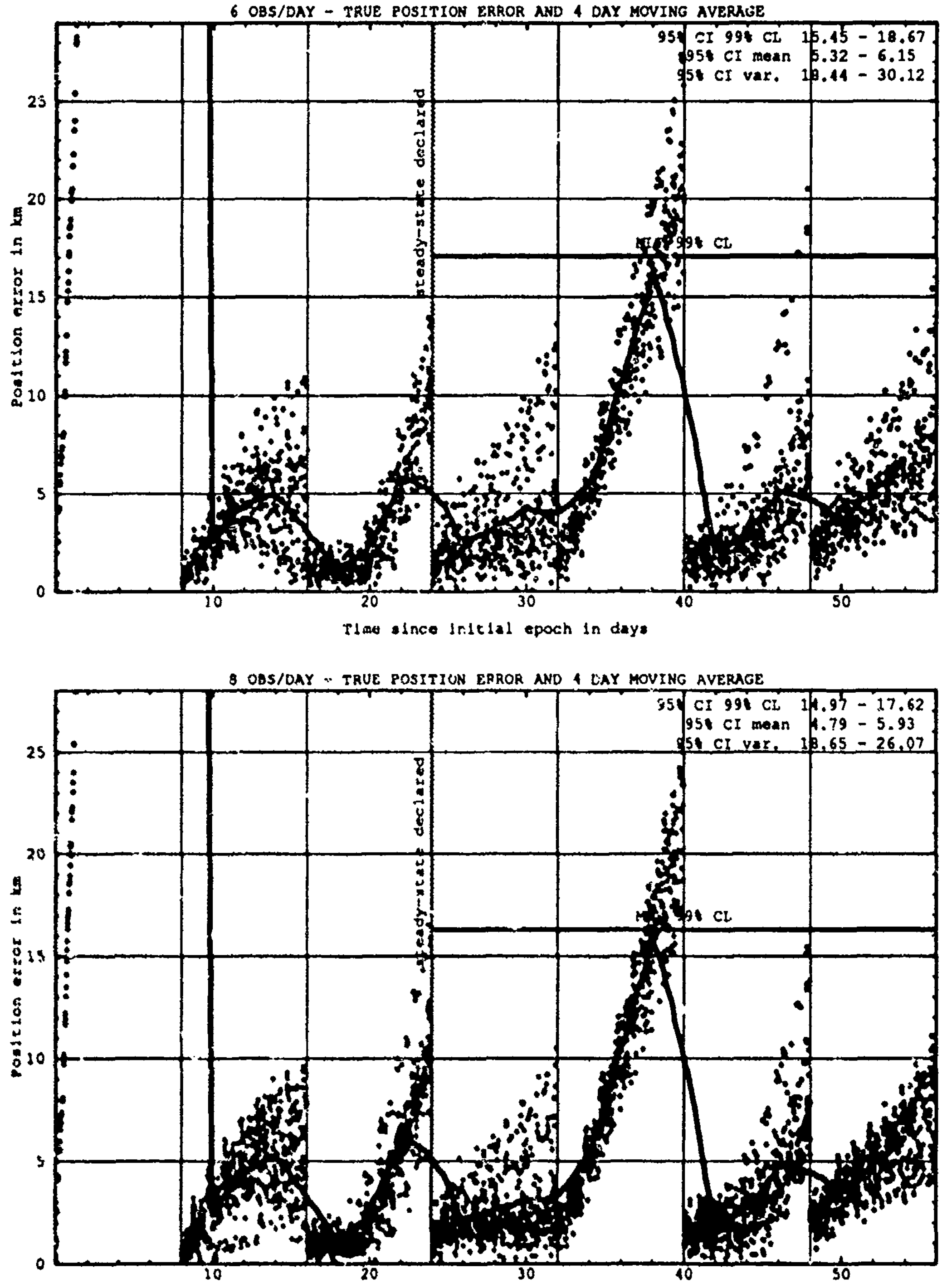


Figure II.89. Class: 3-1-4 (Catalog Number 17429), LUPI 8, OPD 6 and 8.

H.10.2 Differential Corrector Performance. The table below provides statistical information on the last-pass residuals of the differential corrector.

Table H.20. Selected Last-Pass 95% Confidence Interval Statistics.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
17429	8	8	0.95	1.05	0.24	0.33	2.11	2.36

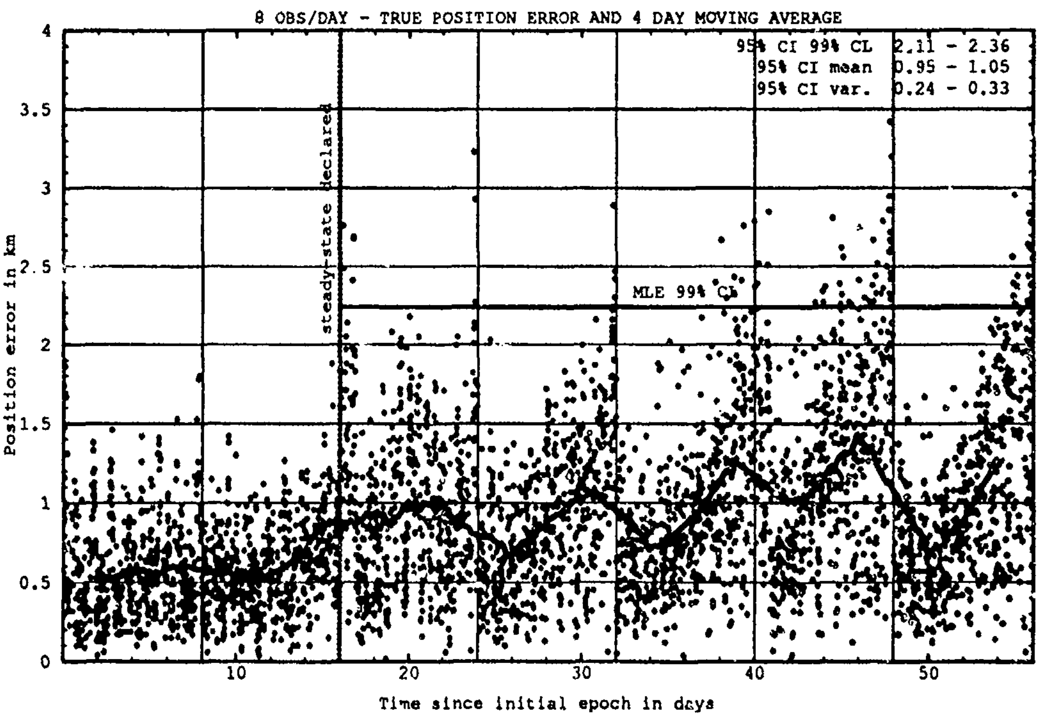


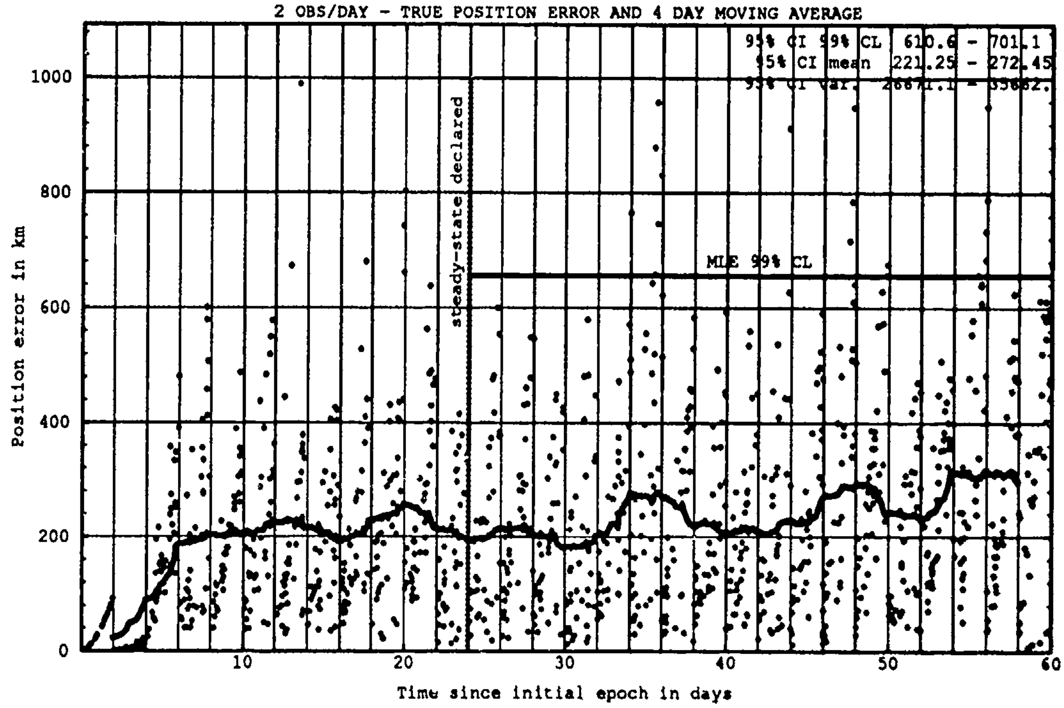
Figure H.90. Last-Pass — Class: 3-1-4 (Catalog Number 17429), LUPI 8, OPD 8.

H.11 CLASS: 4-1-1 (NORAD Catalog Number 20335)

H.11.1 Prediction Performance. The table below provides the statistical information on the ability of the estimates to predict for 1 LUPI after differential correction.

Table H.21. 95% Confidence Interval Analysis on Class: 4–1–1.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
20335	2	2	221.25	272.45	26671.12	35662.20	610.60	701.10
20335	2	4	124.21	159.24	13770.20	18708.94	403.52	470.21
20335	2	6	67.04	81.81	6821.62	11403.47	261.20	326.02
20335	2	8	48.45	59.03	1279.98	4183.47	136.20	203.27



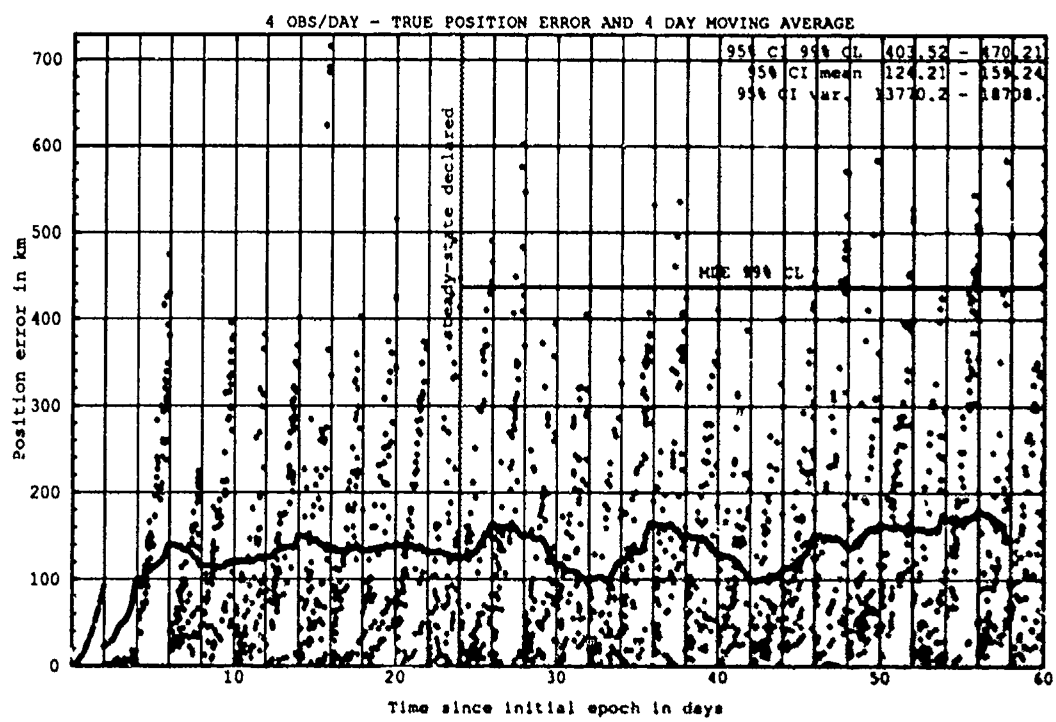
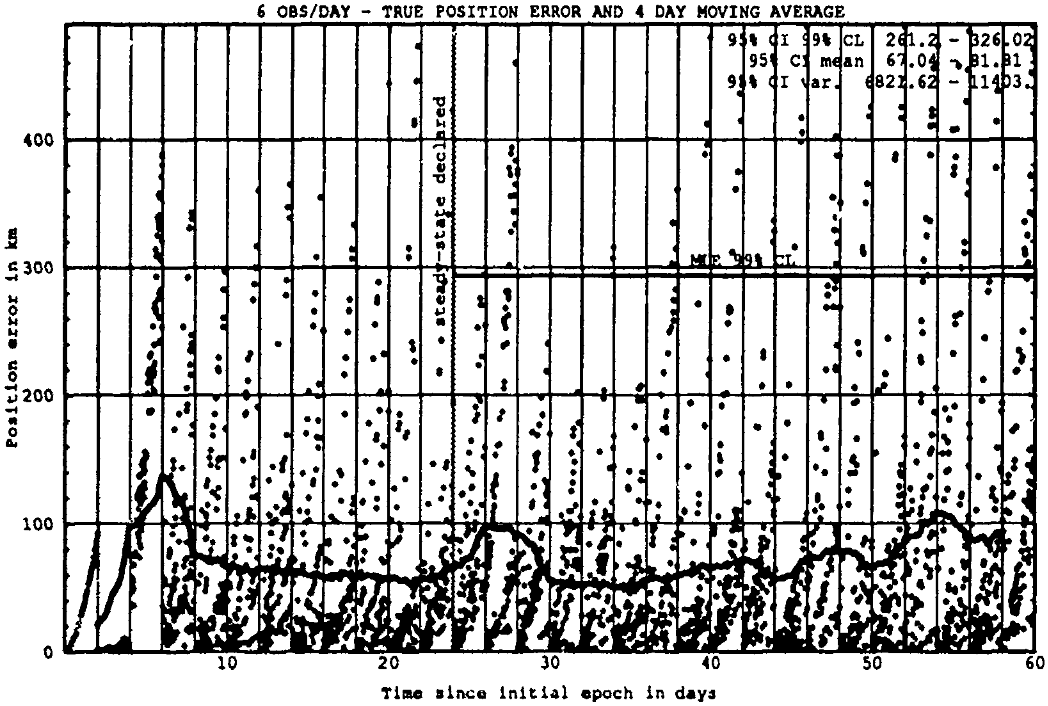


Figure H.91. Class: 4-1-1 (Catalog Number 20335), LUPI 2, OPD 2 and 4.



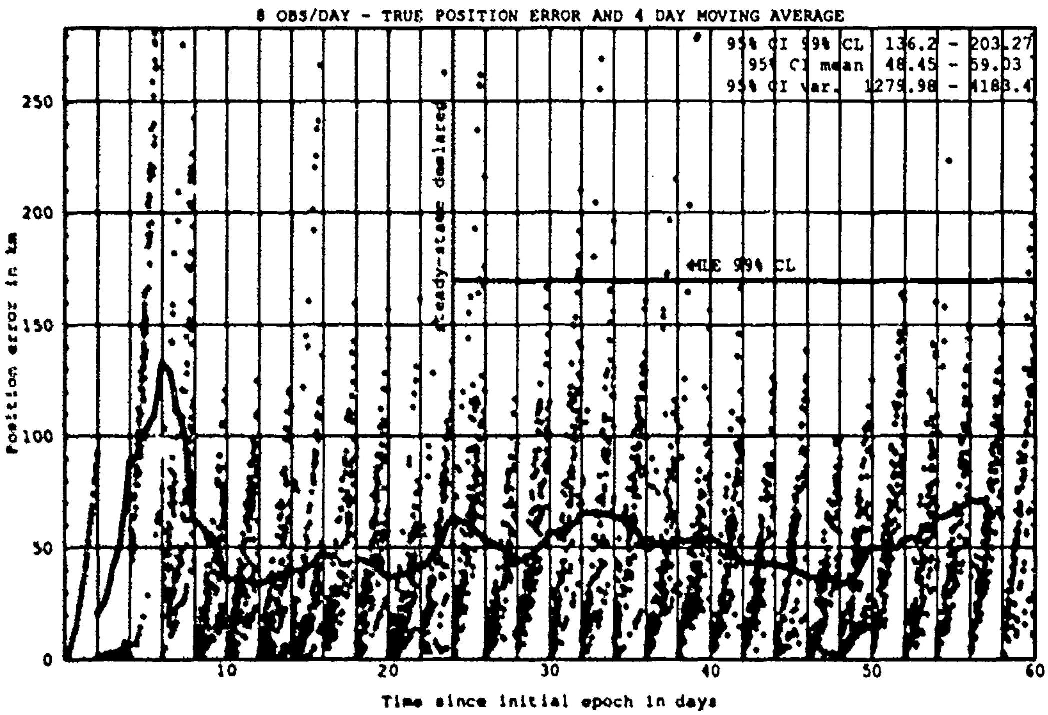


Figure II.92. Class: 4-1-1 (Catalog Number 20335), LUPI 2, OPD 6 and 8.

H.11.2 Differential Corrector Performance. The table below provides statistical information on the last-pass residuals of the differential corrector.

Table H.22. Selected Last-Pass 95% Confidence Interval Statistics.

Catalog Num	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
20335	2	8	4.18	5.99	5.61	36.41	10.70	19.08

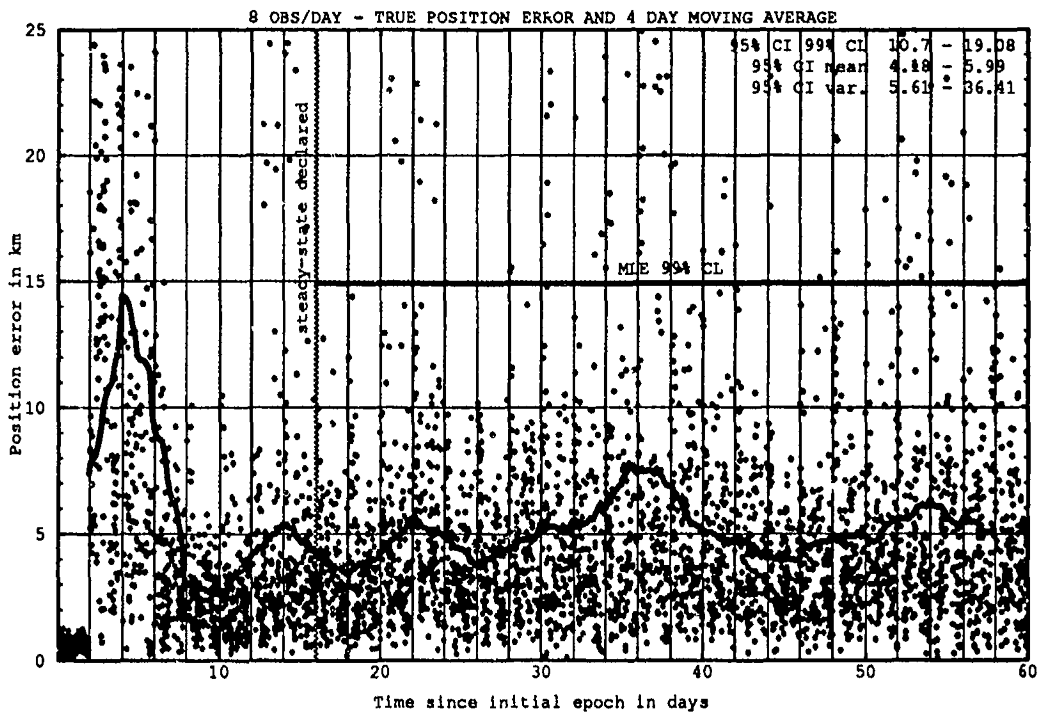


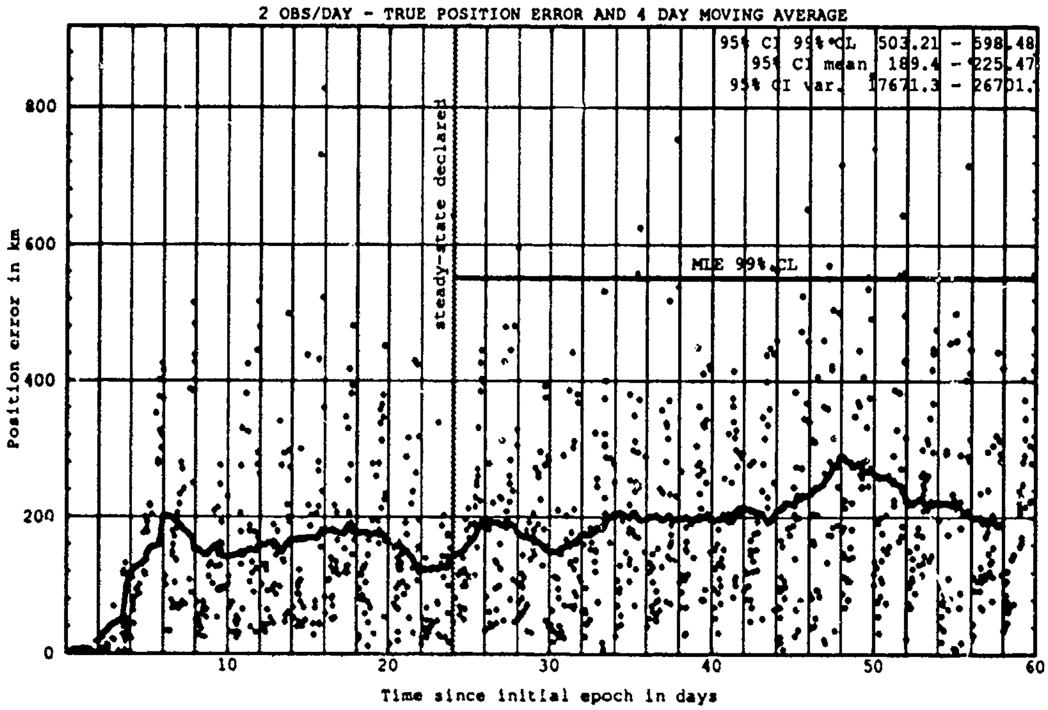
Figure H.93. Last-Pass — Class: 4-1-1 (Catalog Number 20335), LUPI 2, OPD 8.

H.12 CLASS: 4-1-2 (NORAD Catalog Number 15584)

H.12.1 Prediction Performance. The table below provides the statistical information on the ability of the estimates to predict for 1 LUPI after differential correction.

Table H.23. 95% Confidence Interval Analysis on Class: 4-1-2.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
15584	2	2	189.4	225.47	17671.3	26701.23	503.21	598.48
15584	2	4	123.62	161.46	9317.29	13954.9	351.06	432.14
15584	2	6	68.16	88.76	6604.32	9418.69	258.28	312.53
15584	2	8	37.86	60.33	1376.26	5189.10	118.41	224.58



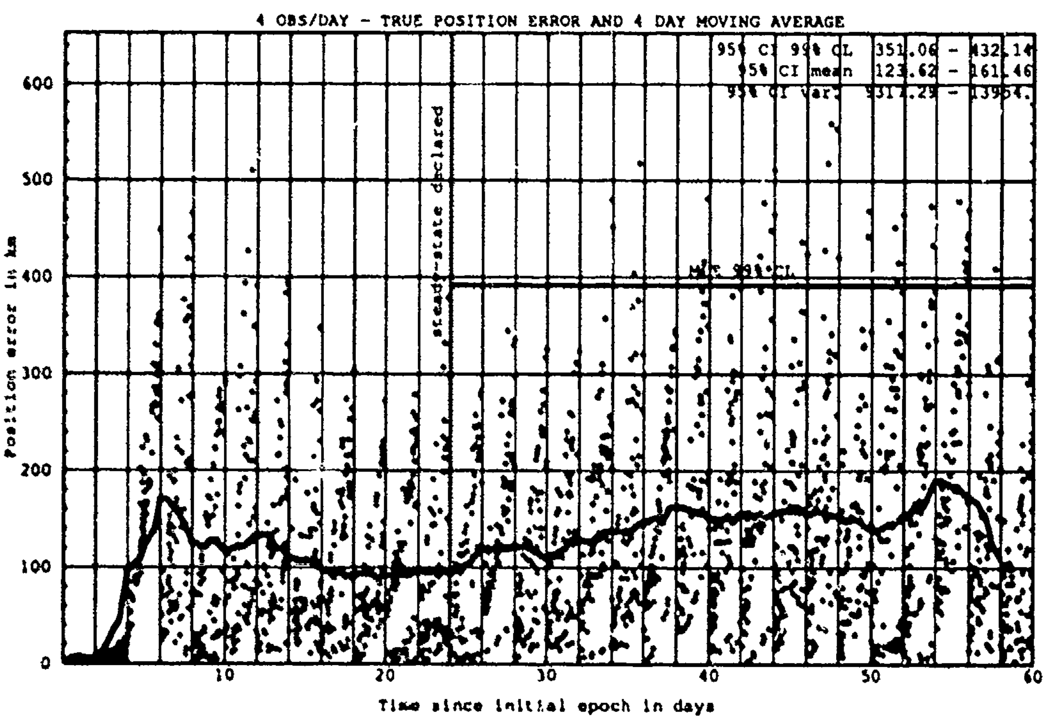
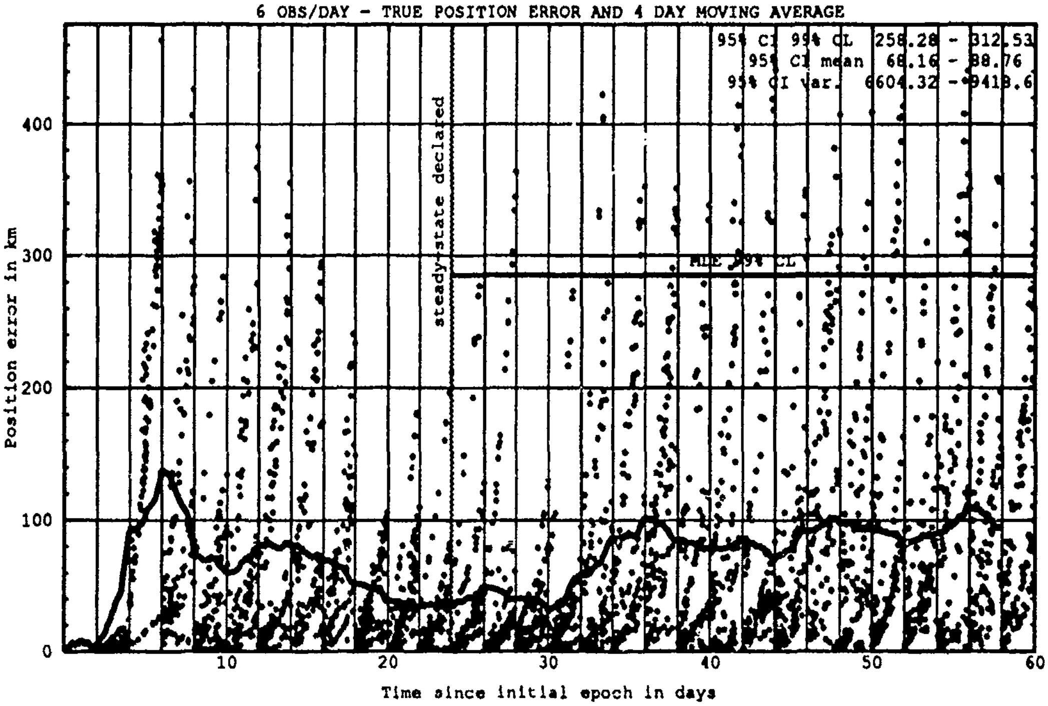


Figure H.94. Class: 4-1-2 (Catalog Number 15584), LUPI 2, OPD 2 and 4.



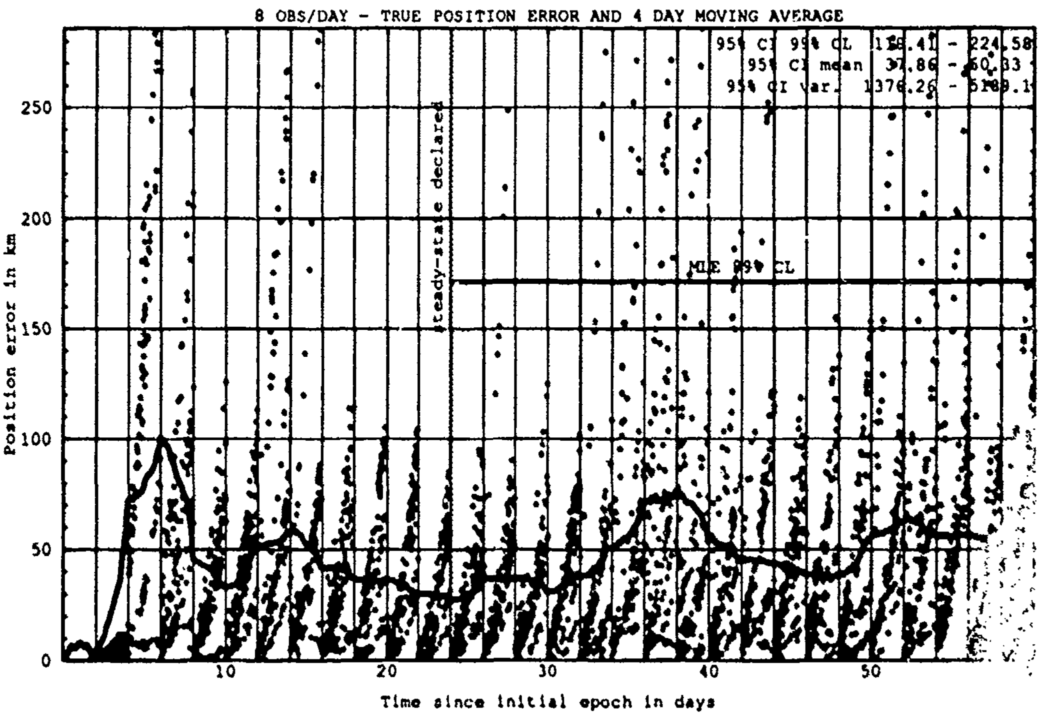


Figure H.95. Class: 4-1-2 (Catalog Number 15584), LUPI 2, OPD 6 and 8.

H.12.2 Differential Corrector Performance. The table below provides statistical information on the last-pass residuals of the differential corrector.

Table H.24. Selected Last-Pass 95% Confidence Interval Statistics.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
15584	2	8	3.98	6.69	NOTE 1	81.52	10.6	25.33

NOTE 1: The 95% confidence interval indicated this value was less than zero, which is not possible.

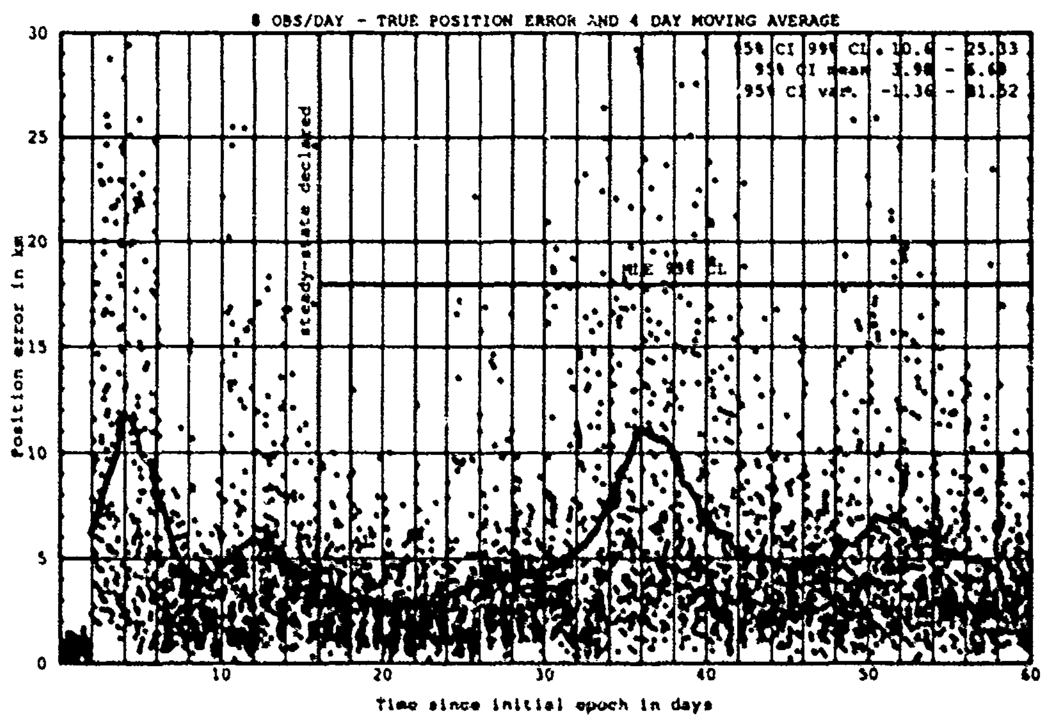


Figure H.96. Last-Pass — Class: 4-1-2 (Catalog Number 15584), LUPI 2, OPD 8.

Appendix I. *TUNED VMAGT GRAPHS*

I.1 CLASS: 1-3-2 (NORAD Catalog Number 14199)

I.1.1 Prediction Performance. The tables below provide statistical information on the ability of estimates to predict for 1 LUPI after tuned differential correction.

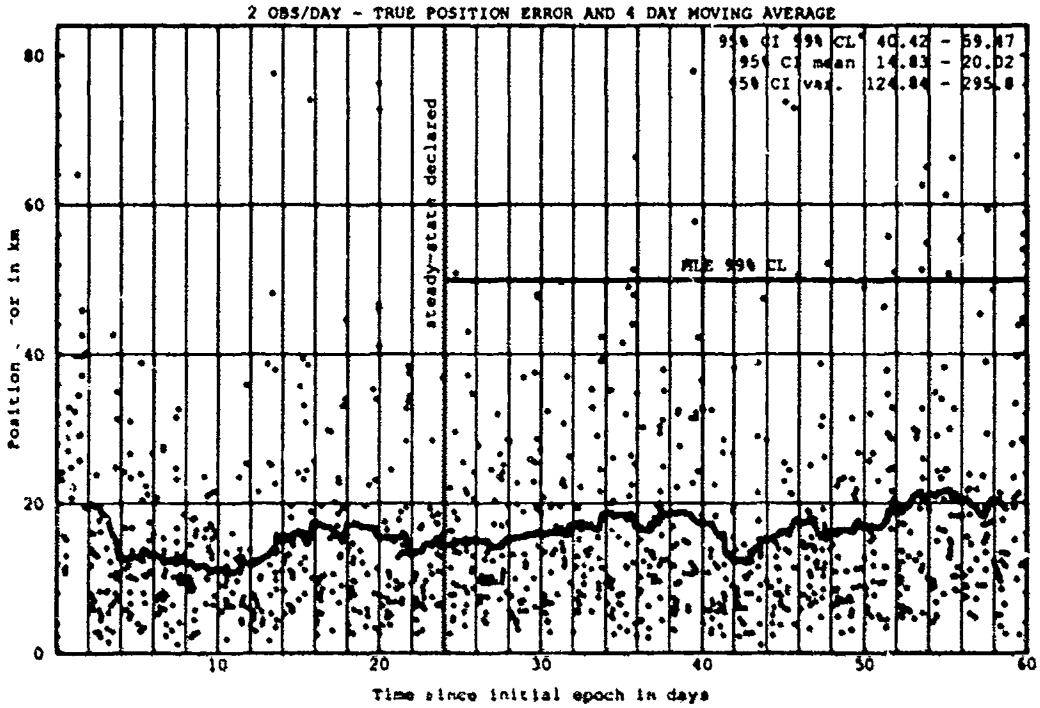
Table I.1. 95% Confidence Interval Analysis on Class: 1-3-2.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
14199	2	2	14.83	20.02	124.84	295.80	40.42	59.47
14199	2	4	9.77	13.85	31.74	135.26	23.72	39.63
14199	2	6	8.89	12.44	NOTE 1	233.04	20.13	43.07
14199	2	8	10.61	12.76	77.51	136.23	30.78	39.78
14199	4	2	16.60	24.67	NOTE 1	1046.81	40.49	90.76
14199	4	4	14.44	19.16	97.08	277.72	38.39	56.40
14199	4	6	11.17	16.19	61.54	205.17	30.62	48.04
14199	4	8	9.51	12.69	47.39	136.55	25.31	39.28
14199	6	2	19.55	26.86	151.66	865.84	53.99	89.57
14199	6	4	8.95	20.66	NOTE 1	474.49	22.60	64.44
14199	6	6	7.65	11.03	20.57	106.35	19.28	33.62
14199	6	8	6.56	8.44	15.27	46.23	15.77	23.81
14199	8	2	11.97	23.97	NOTE 1	722.68	24.98	74.51
14199	8	4	8.07	13.81	0.83	213.65	18.97	44.21
14199	8	6	6.70	9.96	11.71	82.53	16.44	29.57
14199	8	8	6.07	8.18	10.24	46.27	13.48	23.36

NOTE 1: The 95% confidence interval indicated this value was less than zero, which is not possible.

Table I.2. ANOVA Analysis on Class: 1-3-2.

Source of Variation	Degrees of Freedom	Sum of Square	Mean Square	F	Table F
Replications	9	3400.11	377.79	1.24	1.88
Main Effects:					
LUPI	3	5007.16	1669.06	5.49	2.60
OPD	3	25892.25	8630.75	28.42	2.60
Interaction	9	4448.41	494.27	1.63	1.88
Error	135	40993.27	303.65		
Total	159	79741.20			



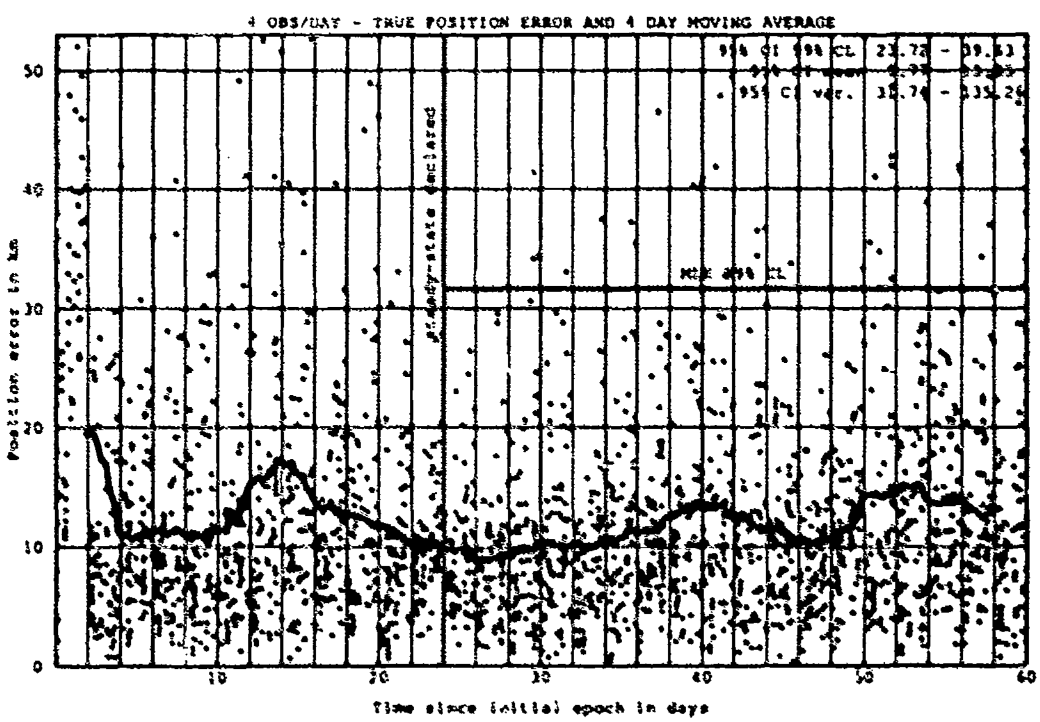
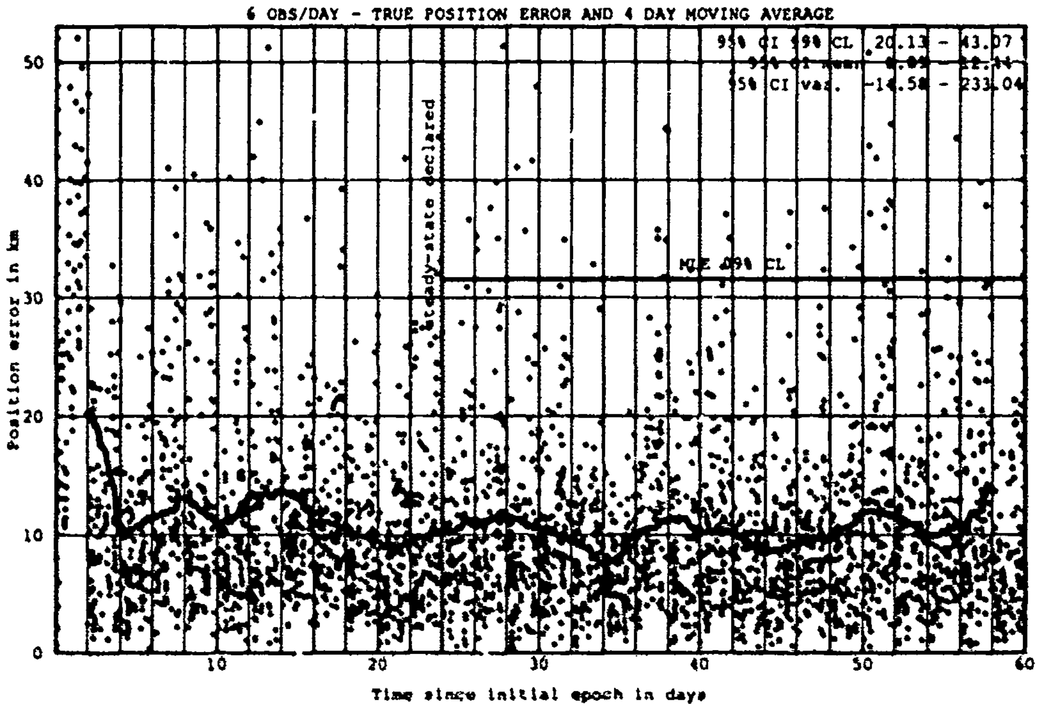


Figure I.1. Class: 1-3-2 (Catalog Number 14199), LUPI 2, OPD 2 and 4.



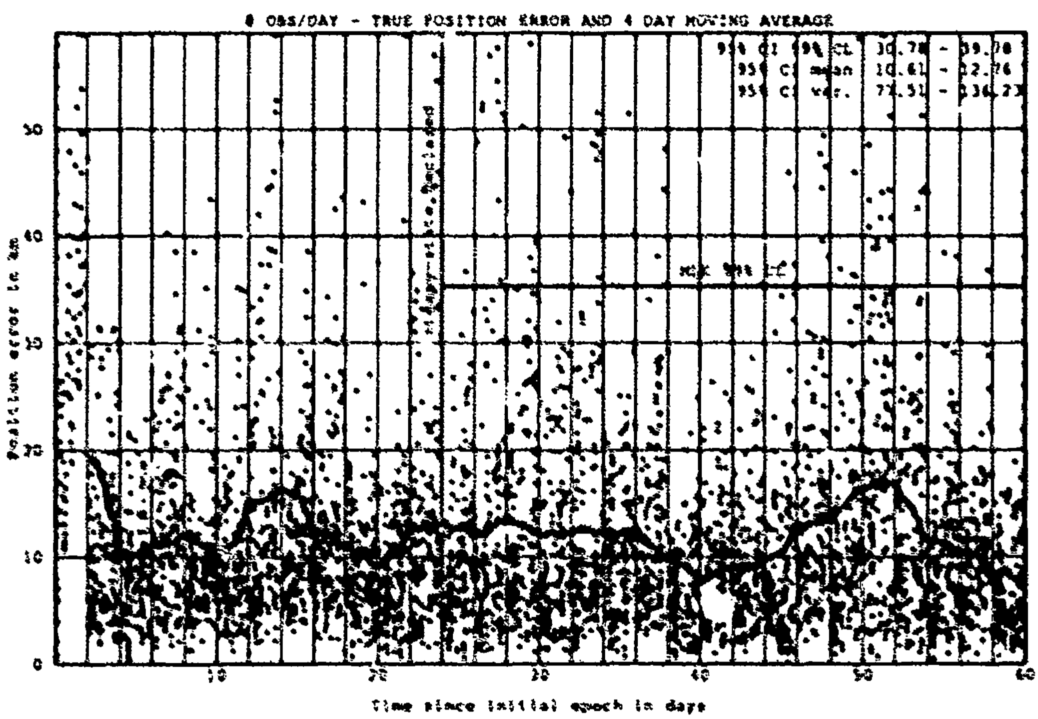
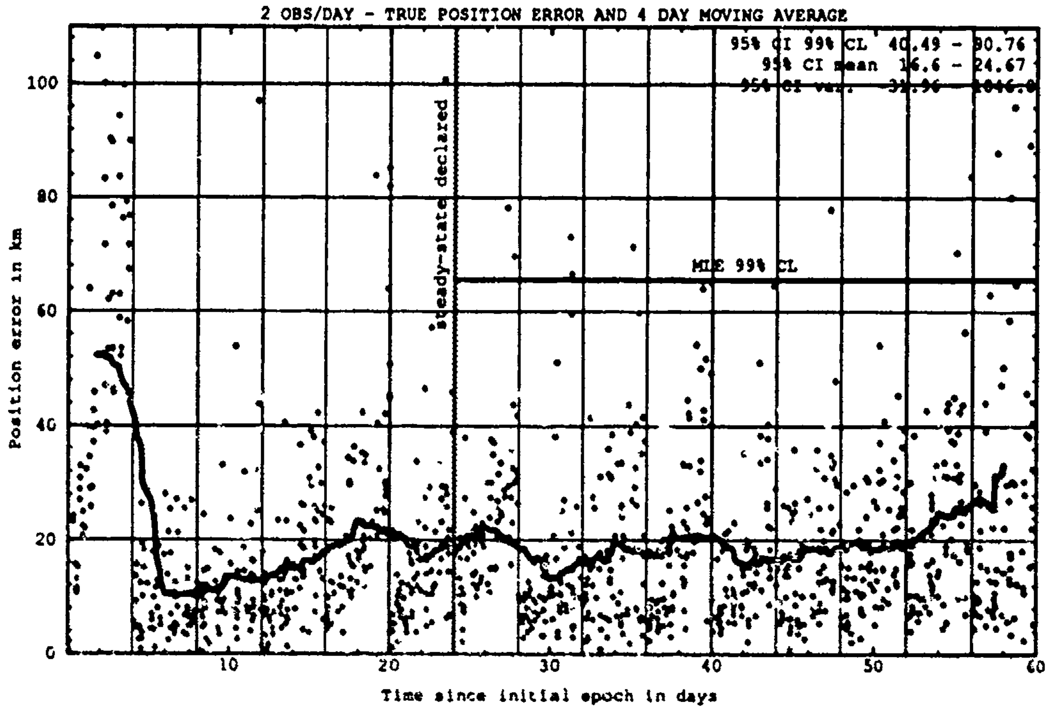


Figure I.2. Class: 1-3-2 (Catalog Number 14199), LUPI 2, OPD 6 and 8.



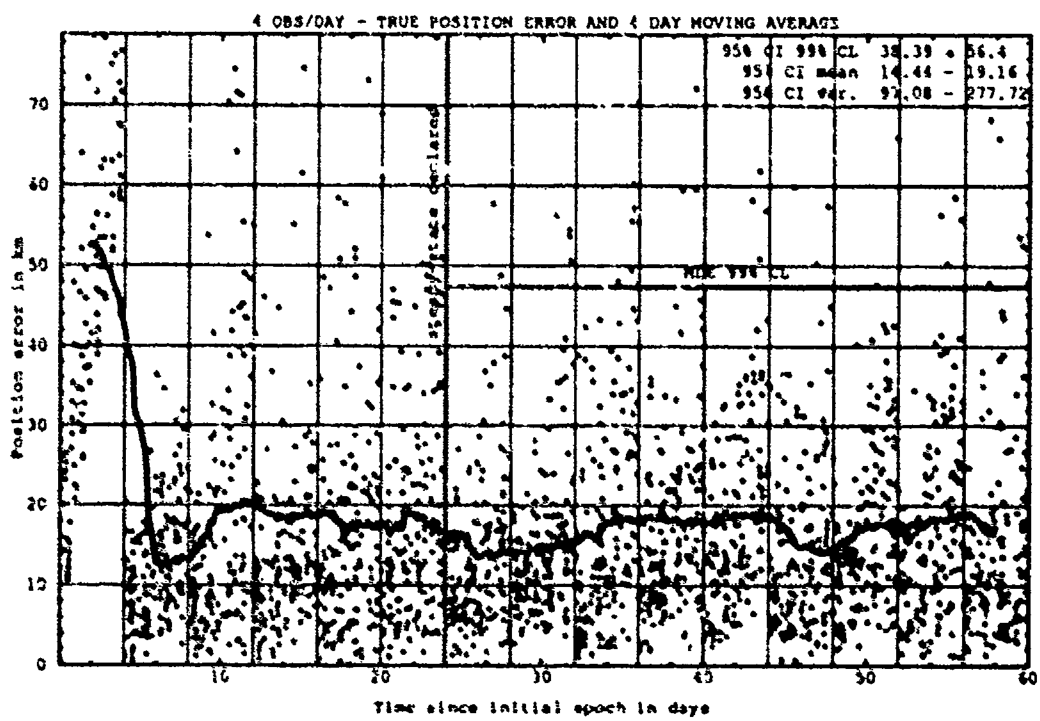
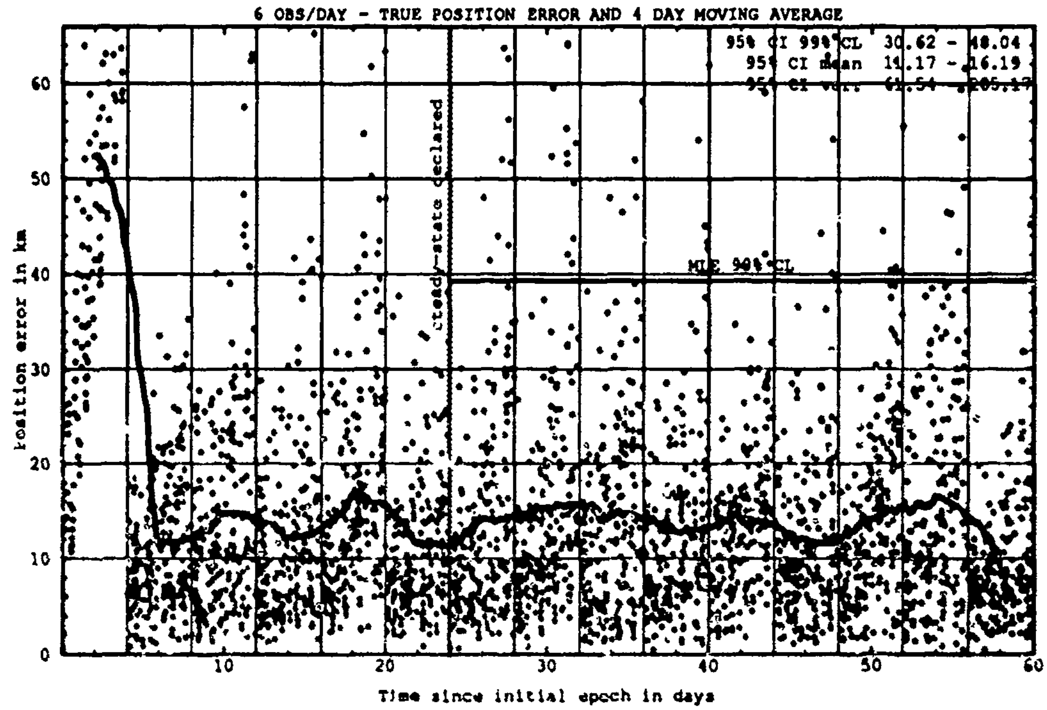


Figure I.3. Class: 1-3-2 (Catalog Number 14199), LUPI 4, OPD 2 and 4.



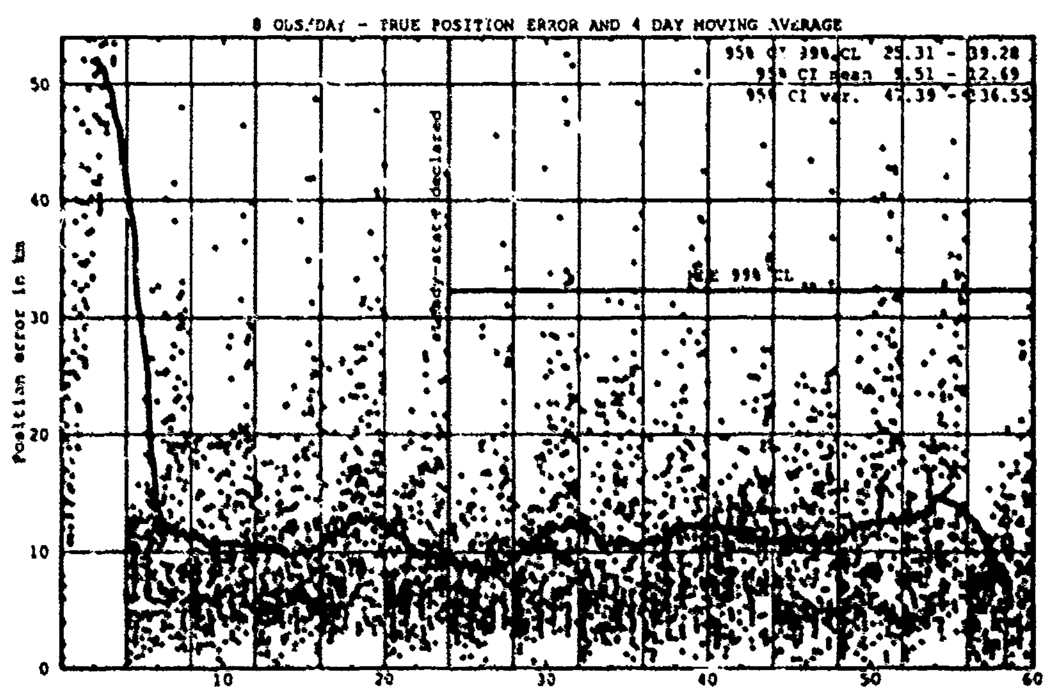
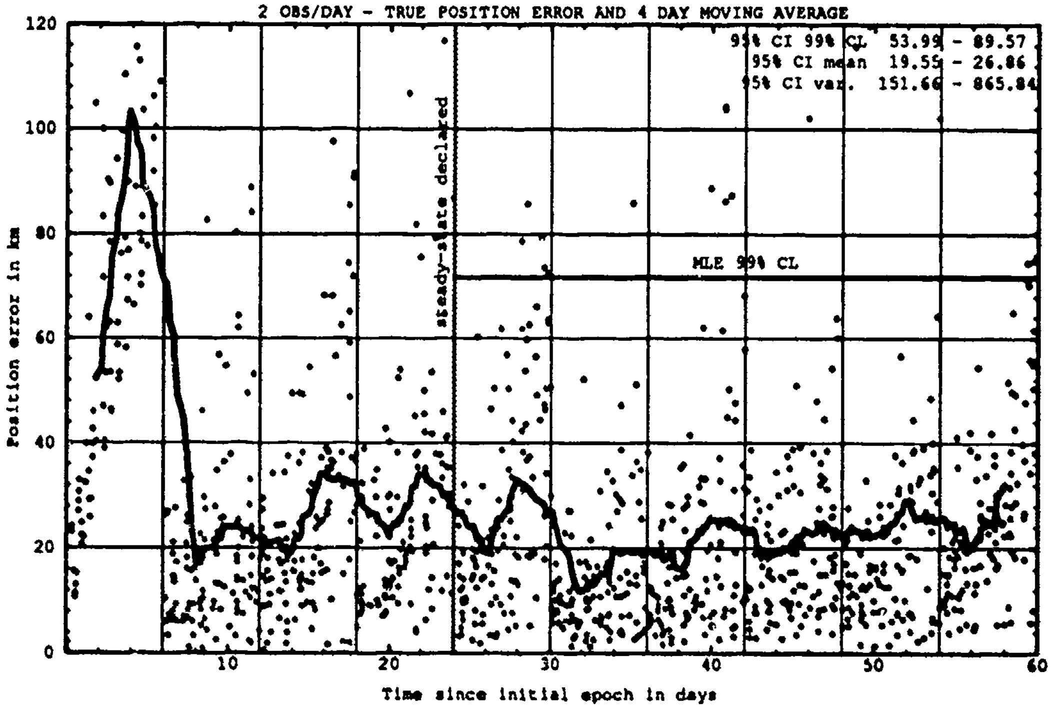


Figure I.4. Class: 1-3-2 (Catalog Number 14199). LUPI 4, OPD 6 and 8.



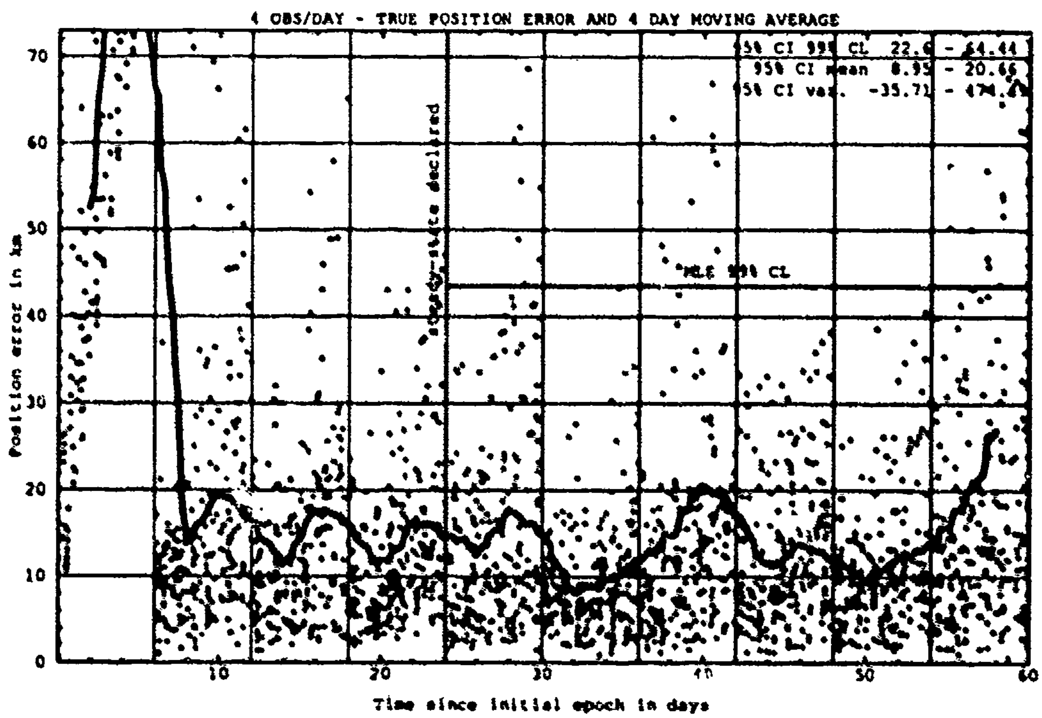
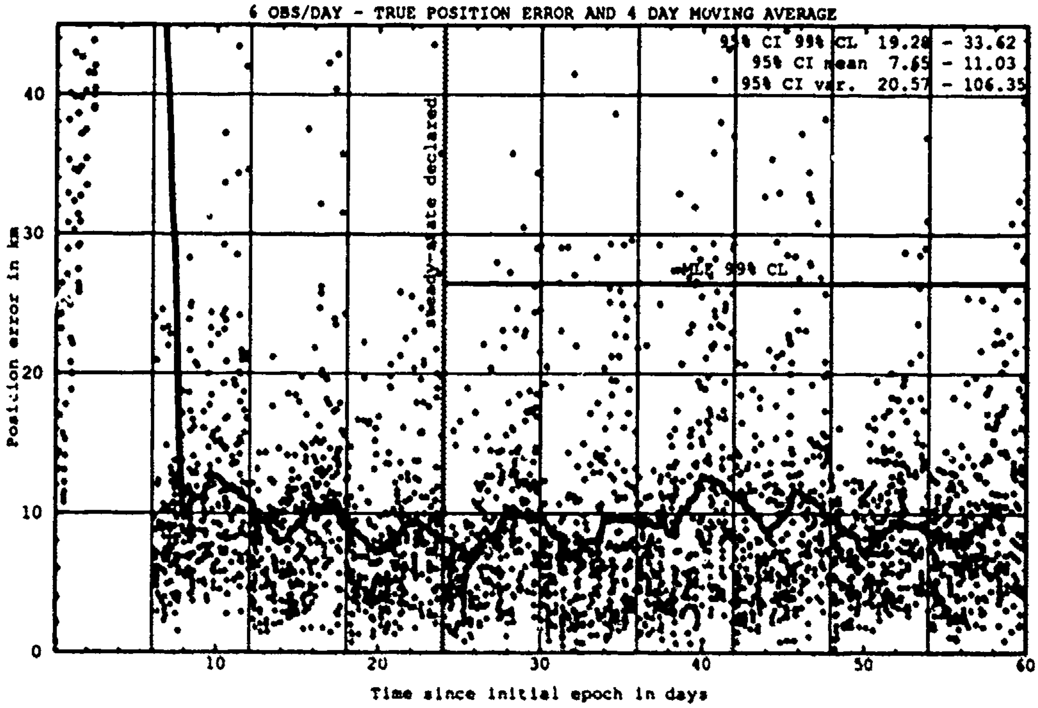


Figure I.5. Class: 1-3-2 (Catalog Number 14199), LUPI 6, OPD 2 and 4.



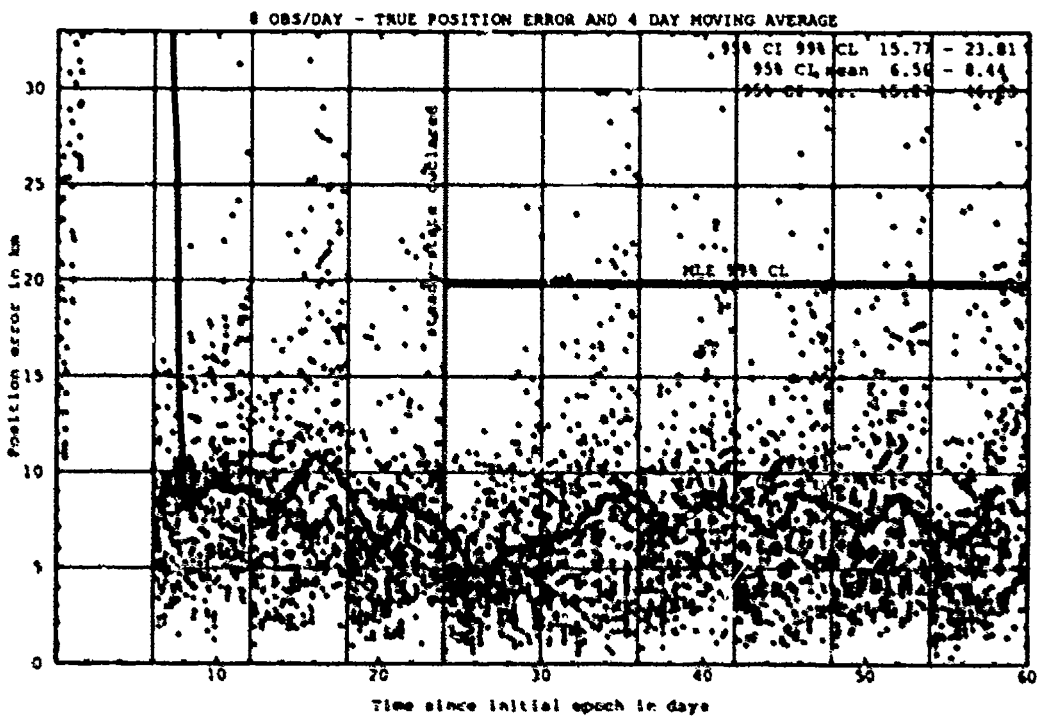
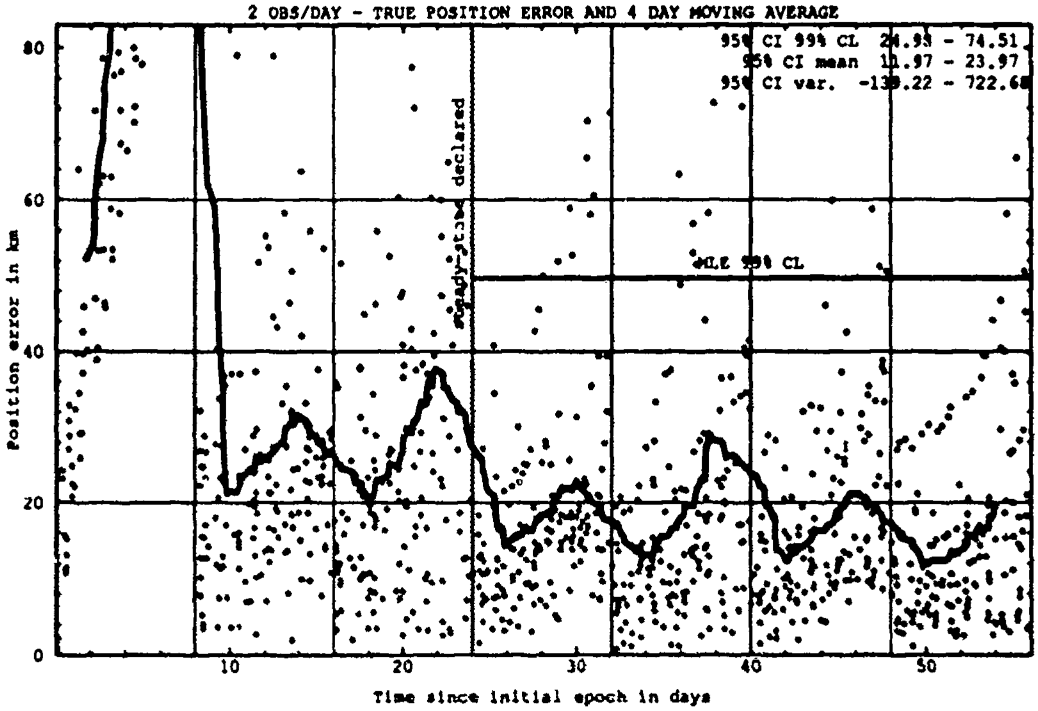


Figure I.6. Class: 1-3-2 (Catalog Number 14199), LUPI 6, OPD 6 and 8.



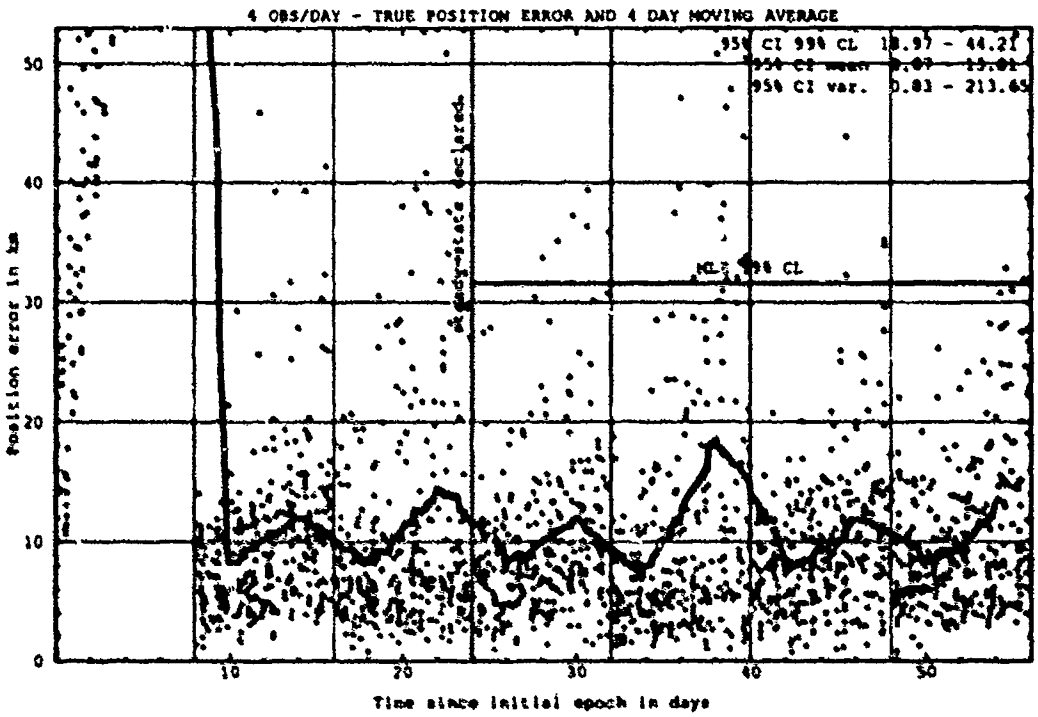


Figure I.7. Class: 1-3-2 (Catalog Number 14199), LUPI 8, OPD 2 and 4.

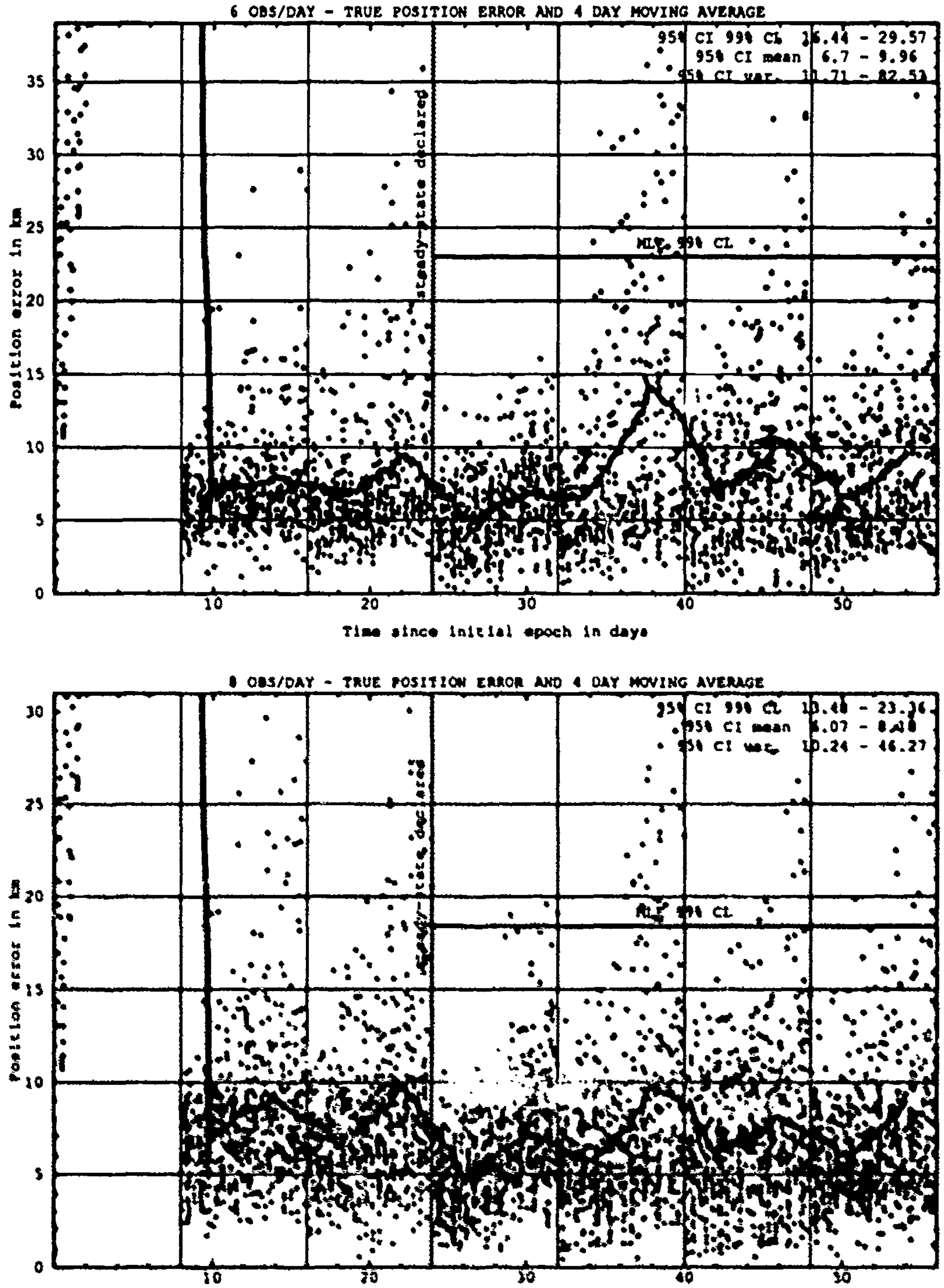


Figure I.6. Class: 1-3-2 (Catalog Number 14199), LUPI 8, OPD 6 and 8.

I.1.2 Differential Corrector Performance. The table below provides statistical information on the last-pass residuals of the tuned differential corrector.

Table I.2. Selected Last-Pass 95% Confidence Interval Statistics.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
14199	8	8	0.87	0.91	0.23	0.25	1.98	2.06

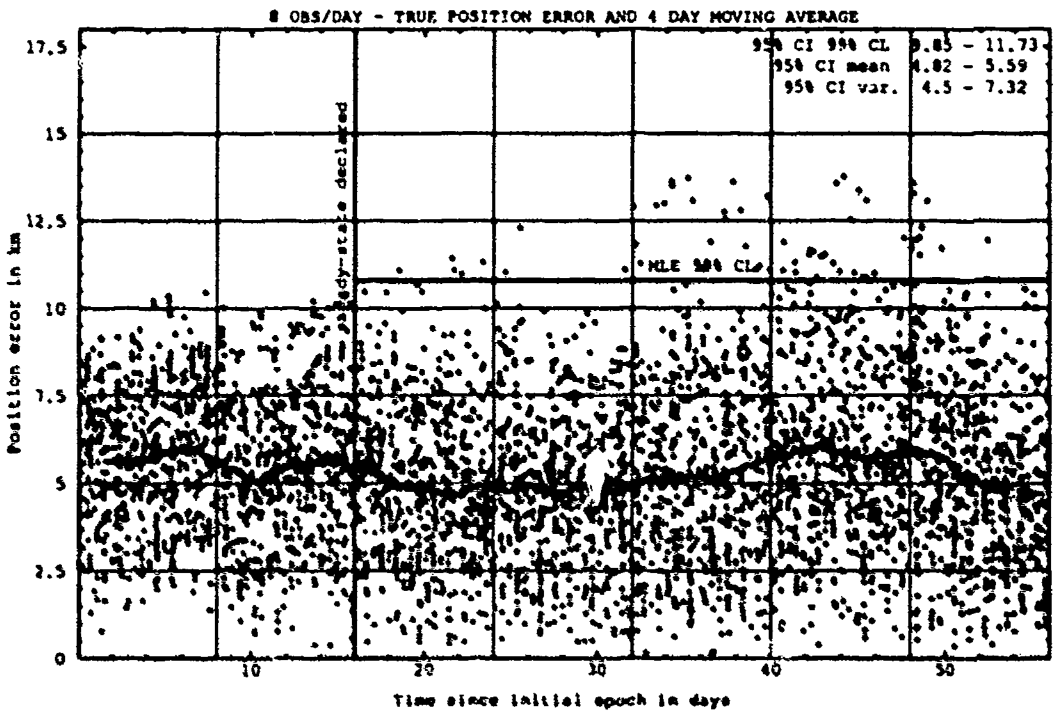


Figure I.9. Last-Pass — Class: 1-3-2 (Catalog Number 14199), LUPI 8, OPD 8.

I.2 CLASS: 3-1-1 (NORAD Catalog Number 01996)

I.2.1 Prediction Performance. The tables below provide the statistical information on the ability of the estimates to predict for 1 LUPI after tuned differential correction.

Table I.4. 95% Confidence Interval Analysis on Class: 3-1-1.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
01996	2	2	2.77	3.63	4.16	9.10	7.61	10.49
01996	2	4	1.99	2.36	2.12	3.22	5.42	6.47
01996	2	6	1.67	2.09	1.23	3.32	4.37	6.19
01996	2	8	1.68	1.98	1.38	2.57	4.42	5.66
01996	4	2	4.57	5.73	17.58	32.87	14.28	18.92
01996	4	4	2.35	2.73	2.11	3.88	5.77	7.24
01996	4	6	1.97	2.36	1.40	2.66	4.73	6.11
01996	4	8	1.89	2.38	1.07	2.60	4.35	6.03
01996	6	2	4.13	6.03	9.89	42.91	11.26	20.72
01996	6	4	2.69	3.61	2.39	5.06	6.37	8.72
01996	6	6	2.49	3.12	2.10	3.89	5.90	7.63
01996	6	8	2.55	3.11	2.57	4.02	6.30	7.72
01996	8	2	4.94	6.42	10.78	29.34	12.78	18.60
01996	8	4	4.39	5.64	9.51	18.08	11.56	15.41
01996	8	6	4.18	5.31	10.43	17.00	11.67	14.83
01996	8	8	4.17	5.12	10.06	15.65	11.55	14.26

Table I.5. ANOVA Analysis on Class: 3-1-1.

Source of Variation	Degrees of Freedom	Sum of Square	Mean Square	F	Table F
Replications	9	82.19	9.13	1.66	1.88
Main Effects:					
LUPI	3	1201.65	400.55	72.68	2.60
OPD	3	1271.22	423.74	76.89	2.60
Interaction	9	373.78	41.53	7.53	1.88
Error	135	734.96	5.51		
Total	159	3672.81			

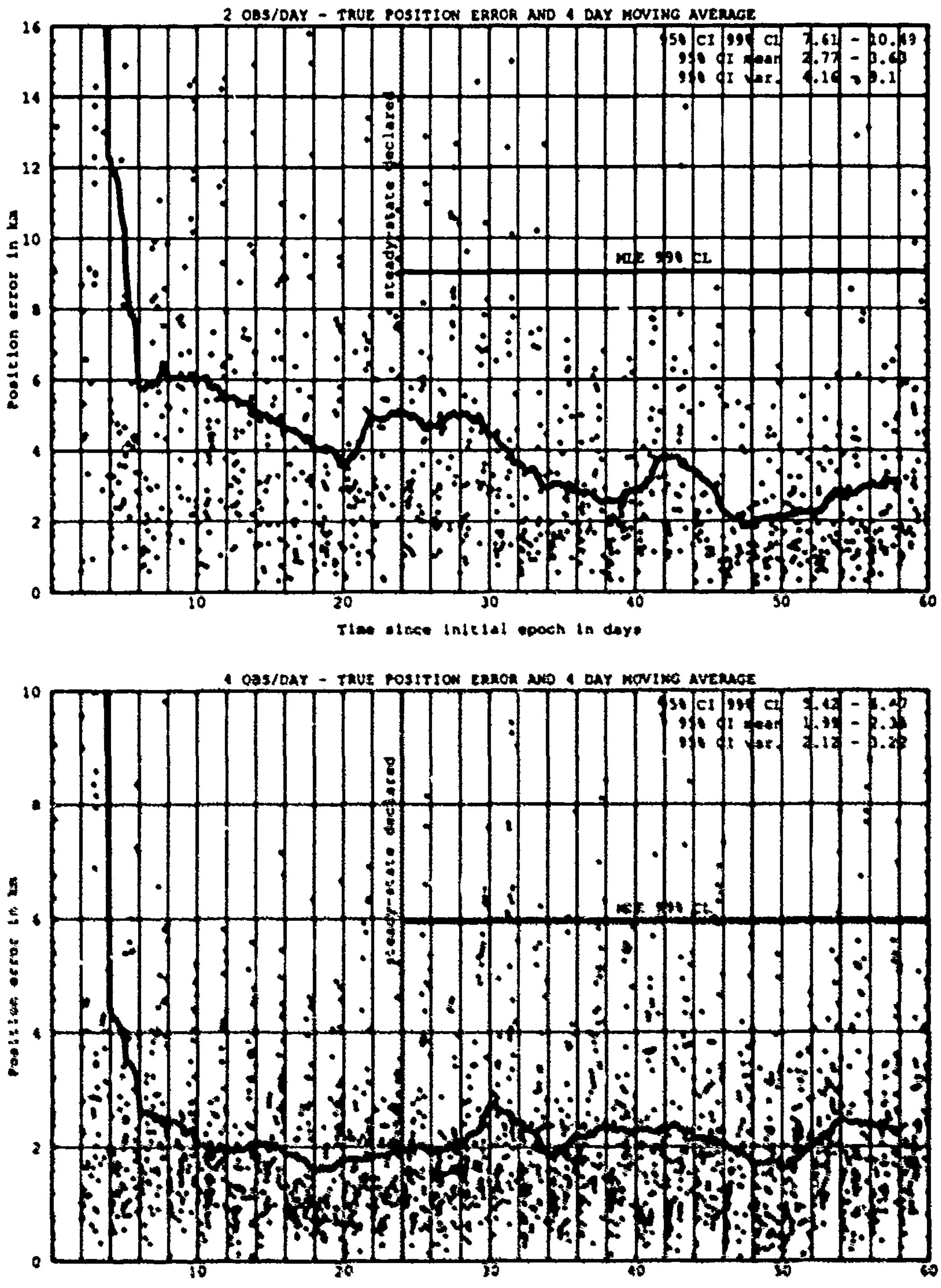


Figure I.10. Class: 3-1-1 (Catalog Number 01996), LUPI 2, OPD 2 and 4.

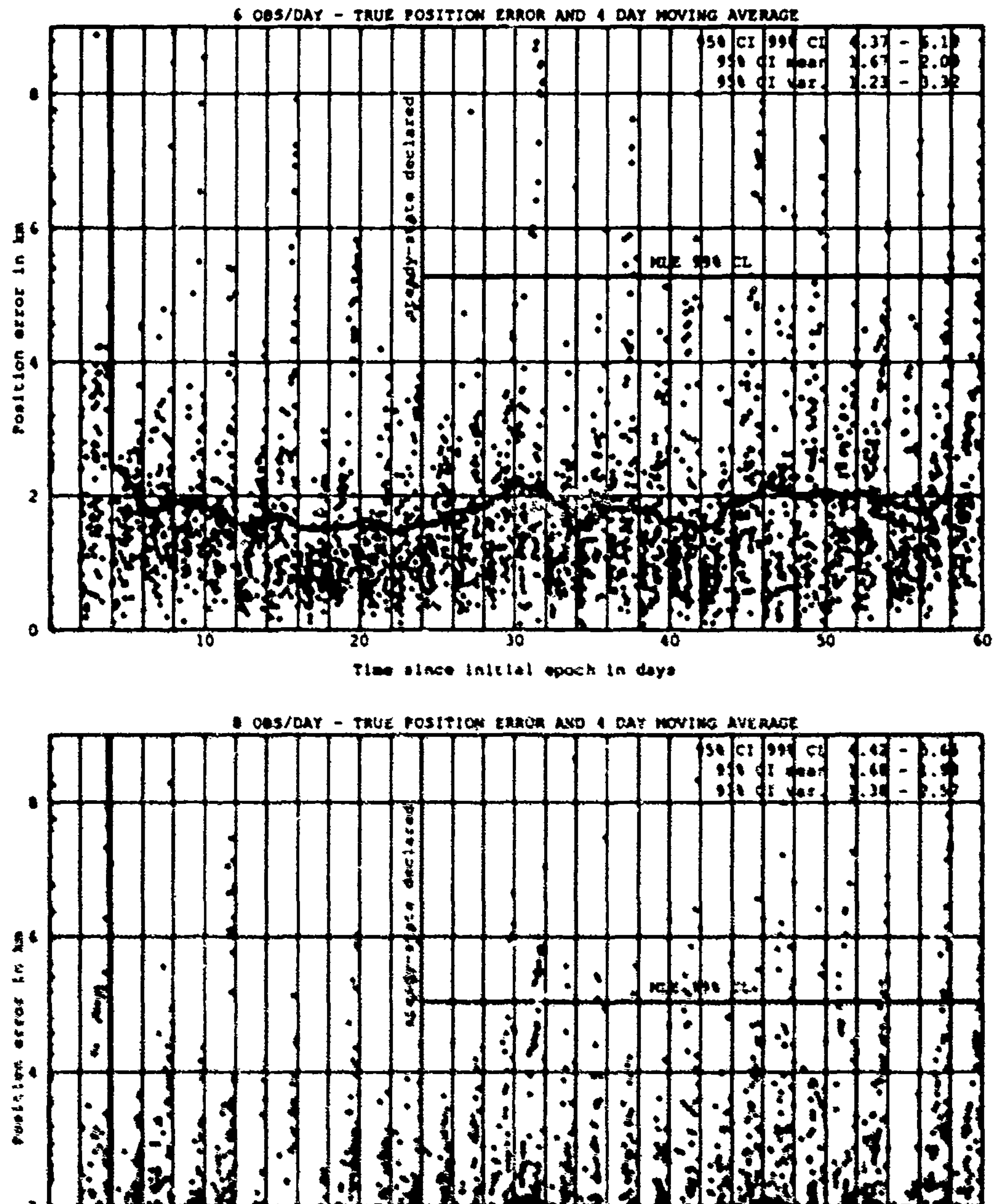


Figure I.11. Class: 3-1-1 (Catalog Number 01996), LUPI 2, OPD 6 and 8.

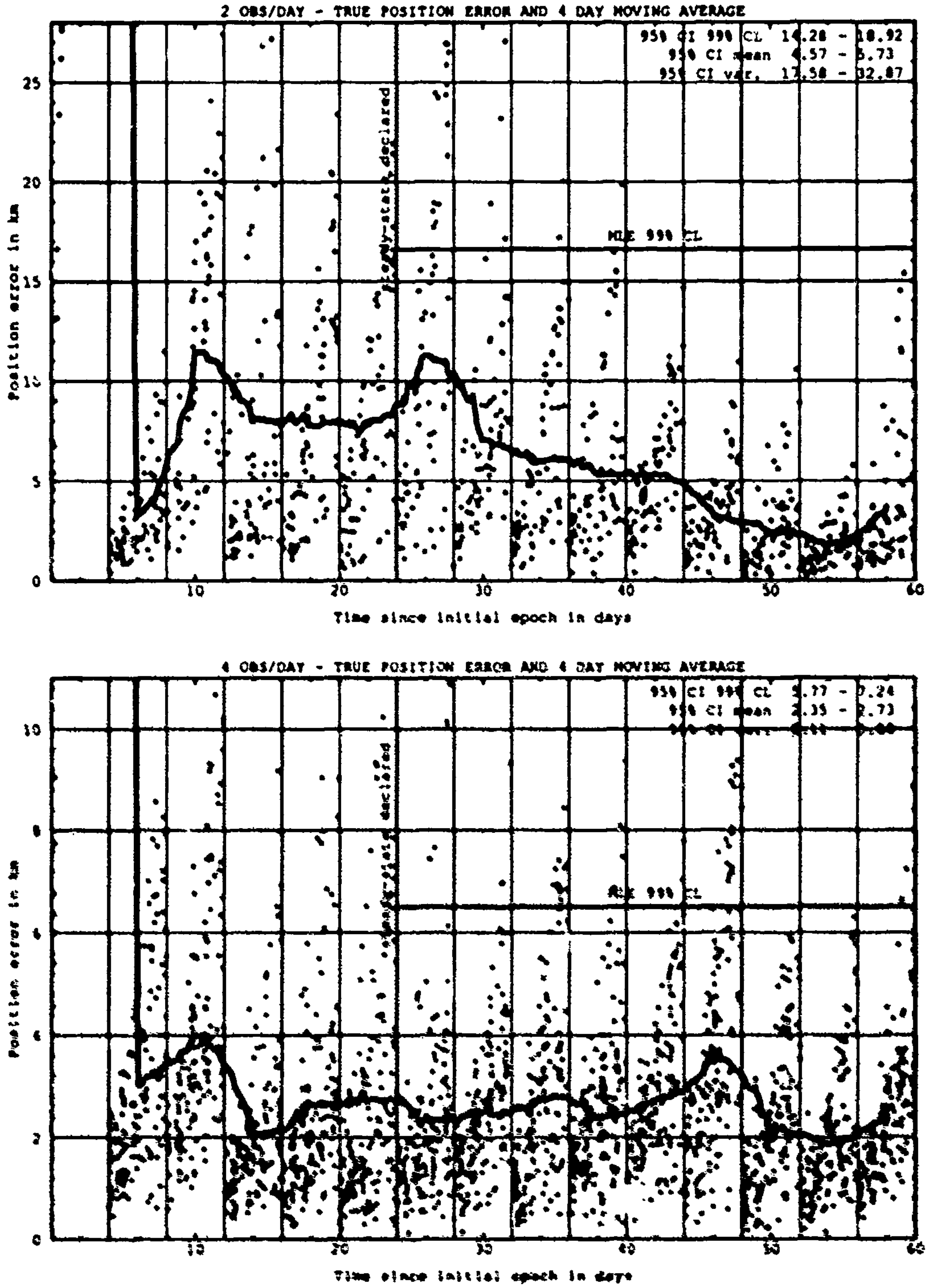


Figure I.12. Class: 3-1-1 (Catalog Number 01996), LUPI 4, OPD 2 and 4.

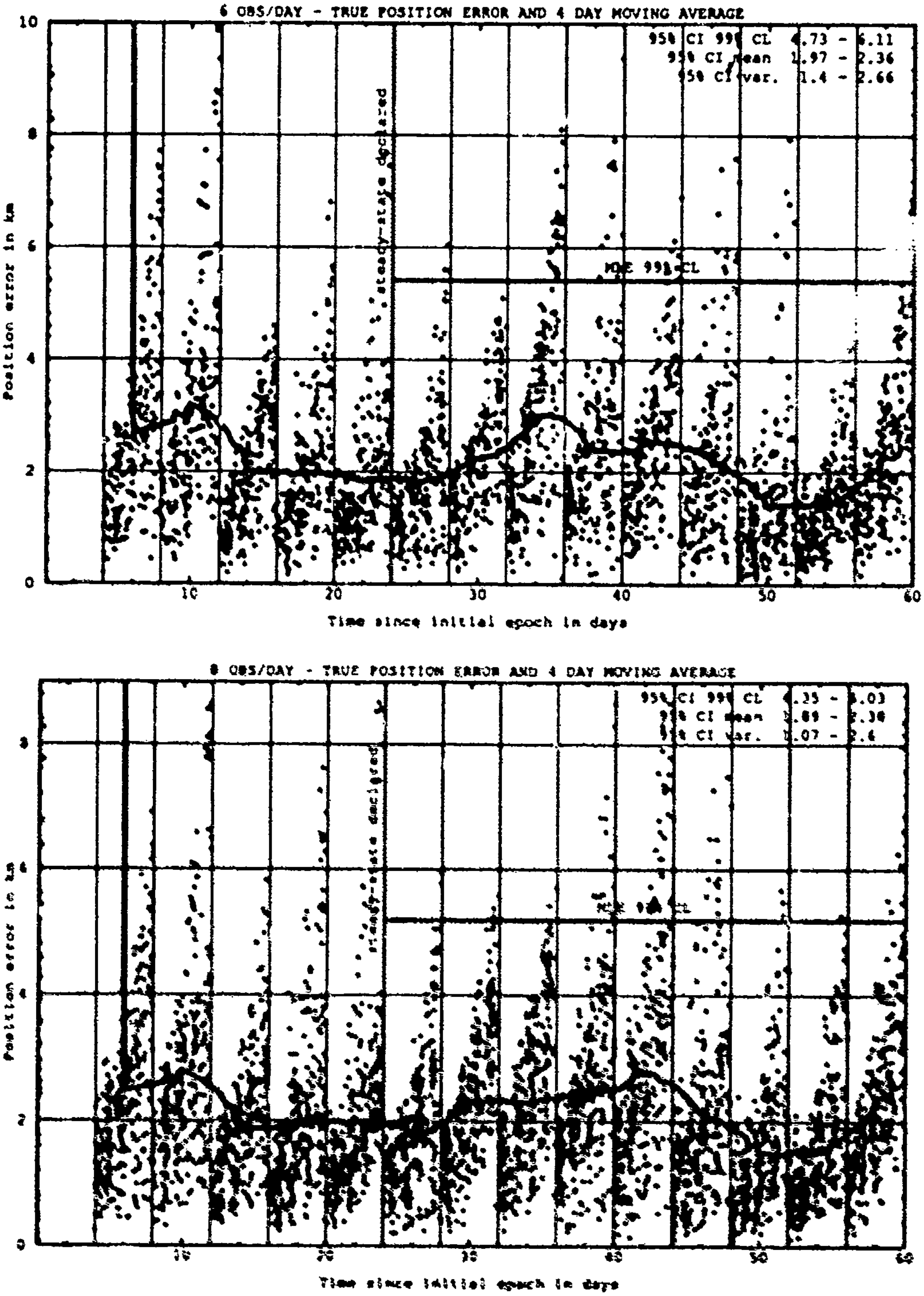
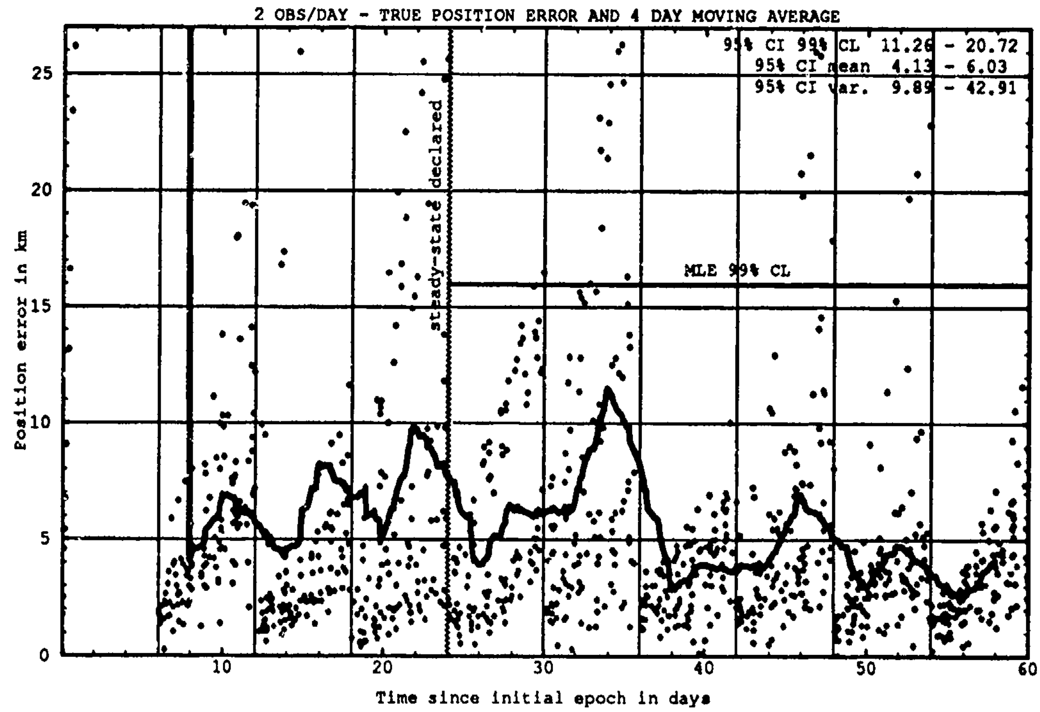


Figure I.13. Class: 3-1-1 (Catalog Number 01990), LUPI 4, OPD 6 and 8.



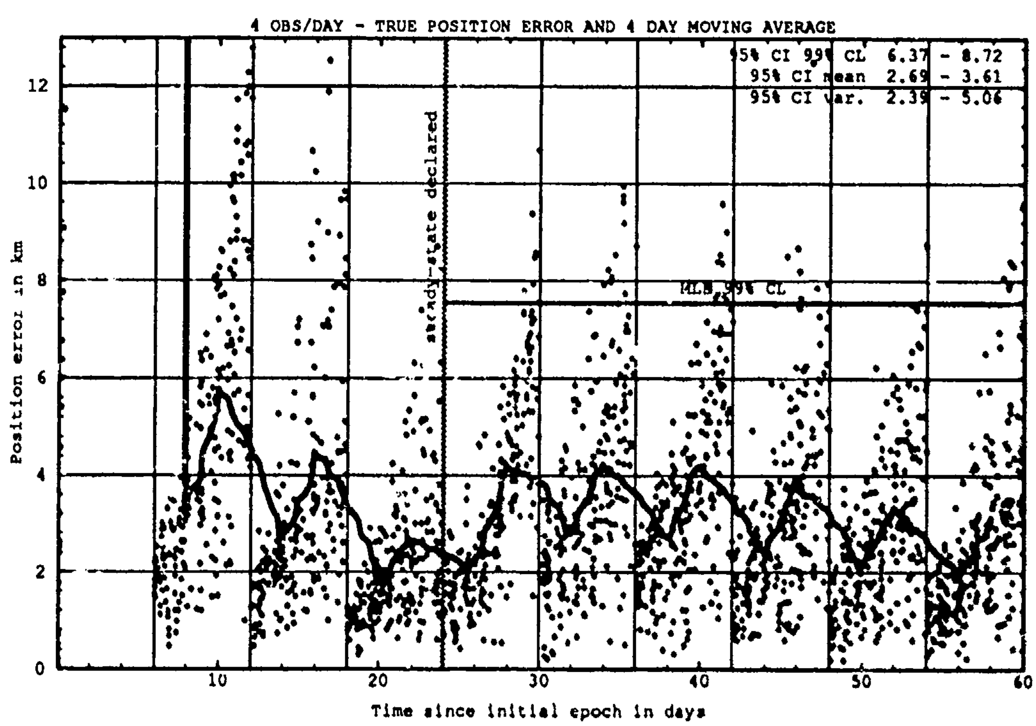
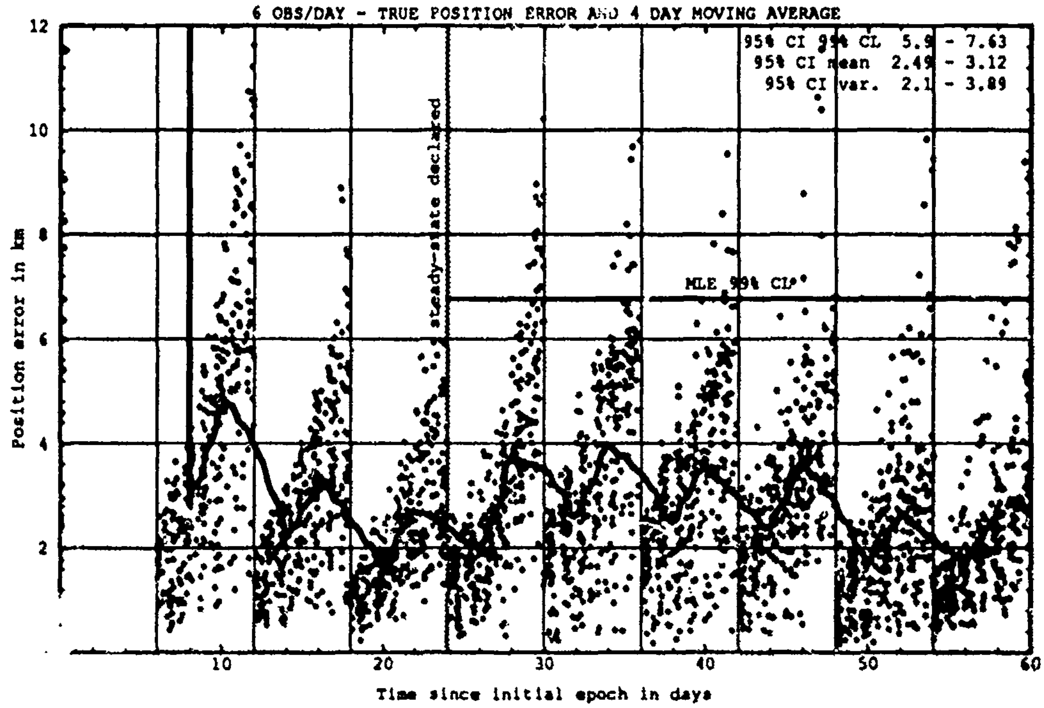


Figure I.14. Class: 3-1-1 (Catalog Number 01996), LUPI 6, OPD 2 and 4.



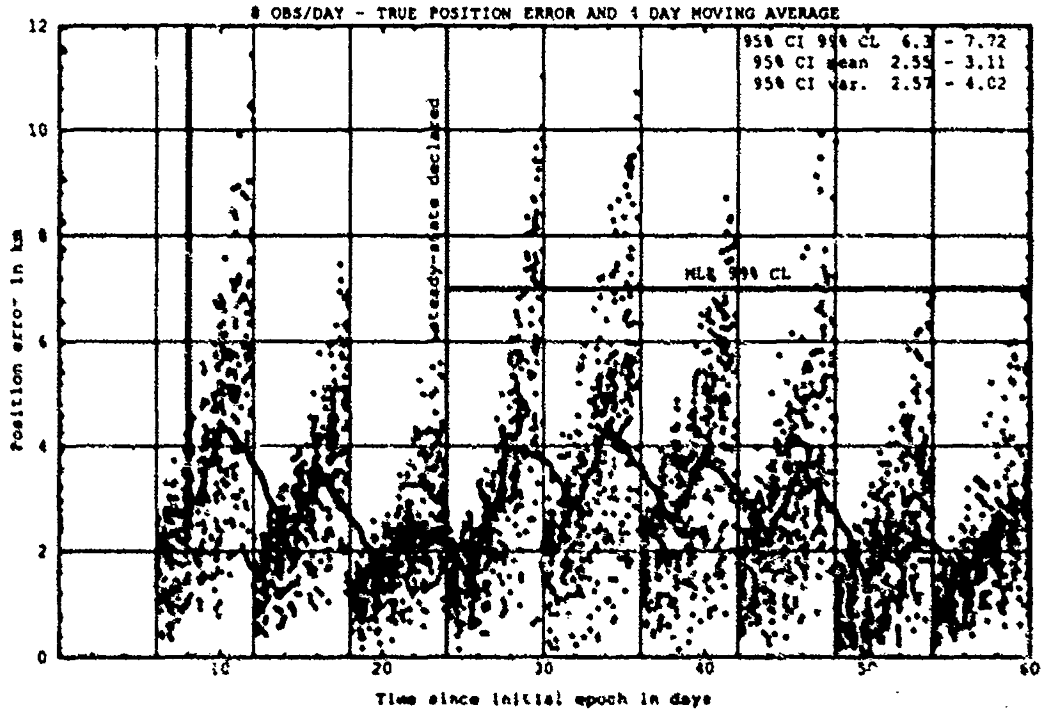


Figure I.15. Class: 3-1-1 (Catalog Number 01996), LUPI 6, OPD 6 and 8.

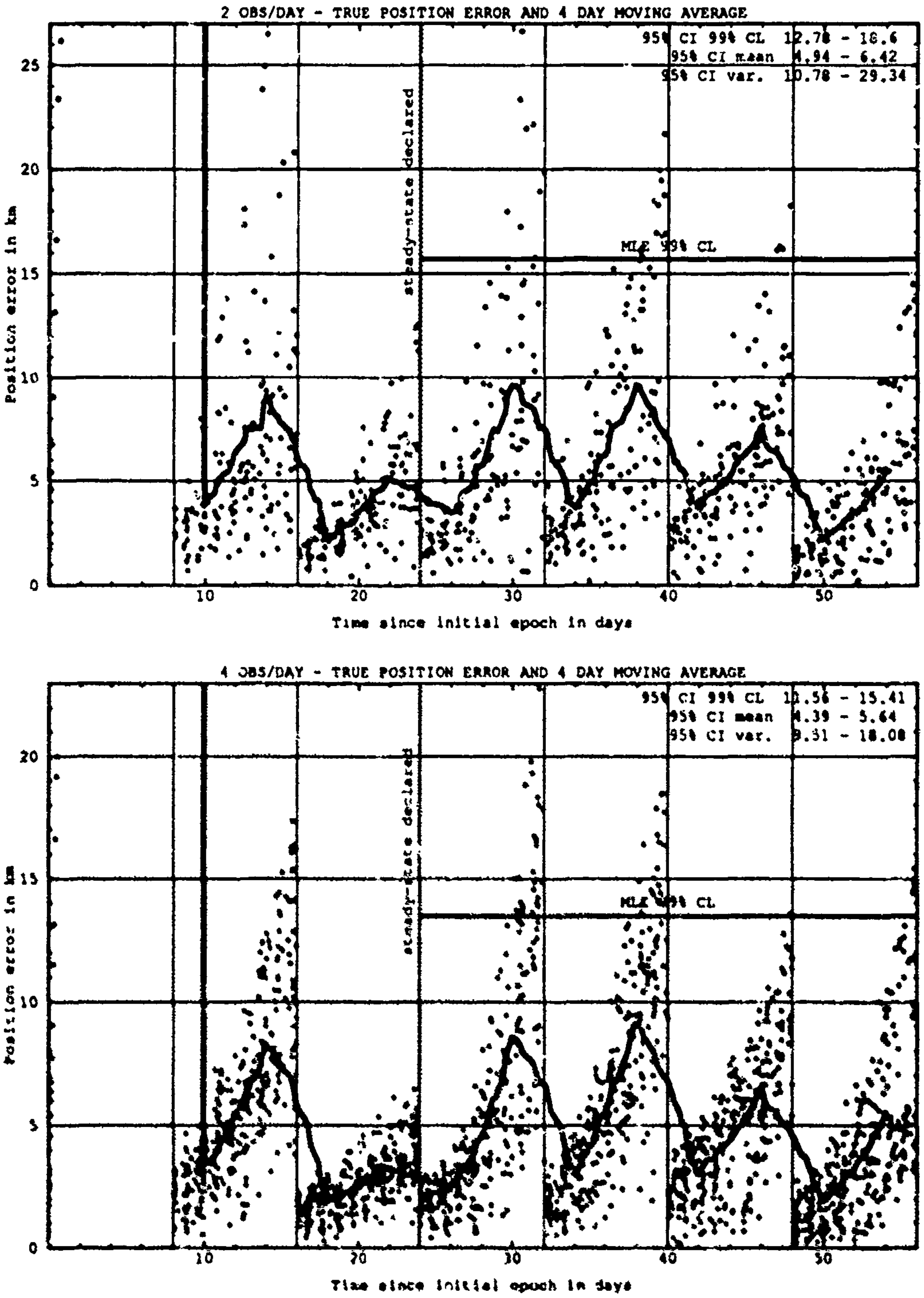
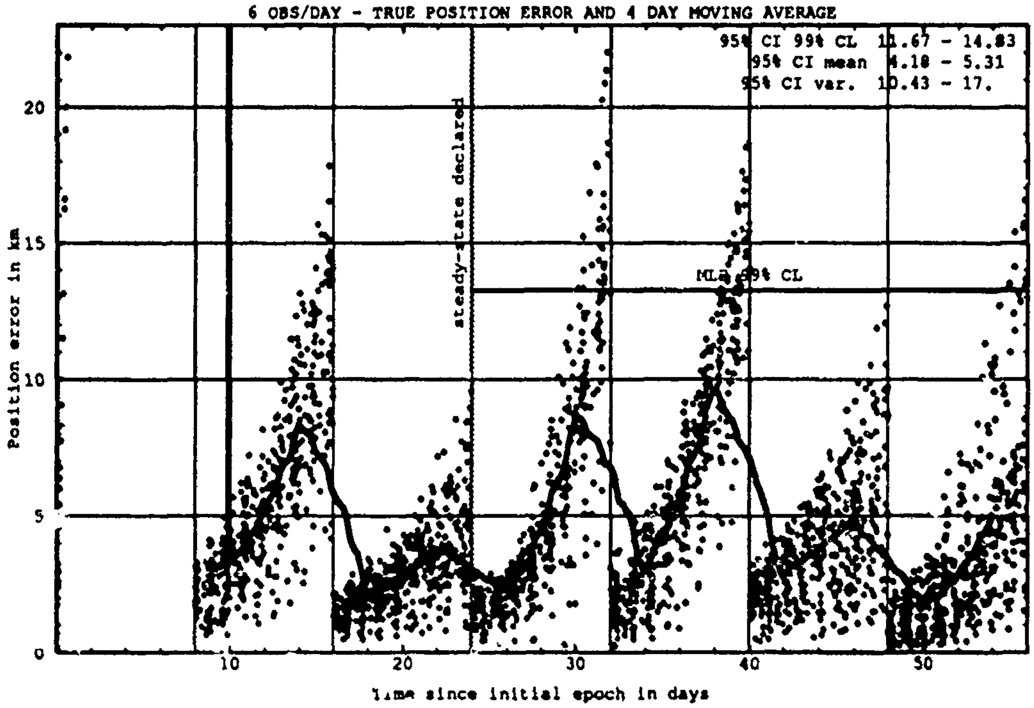


Figure I.16. Class: 3-1-1 (Catalog Number 01996), LUPI 8, OPD 2 and 4.



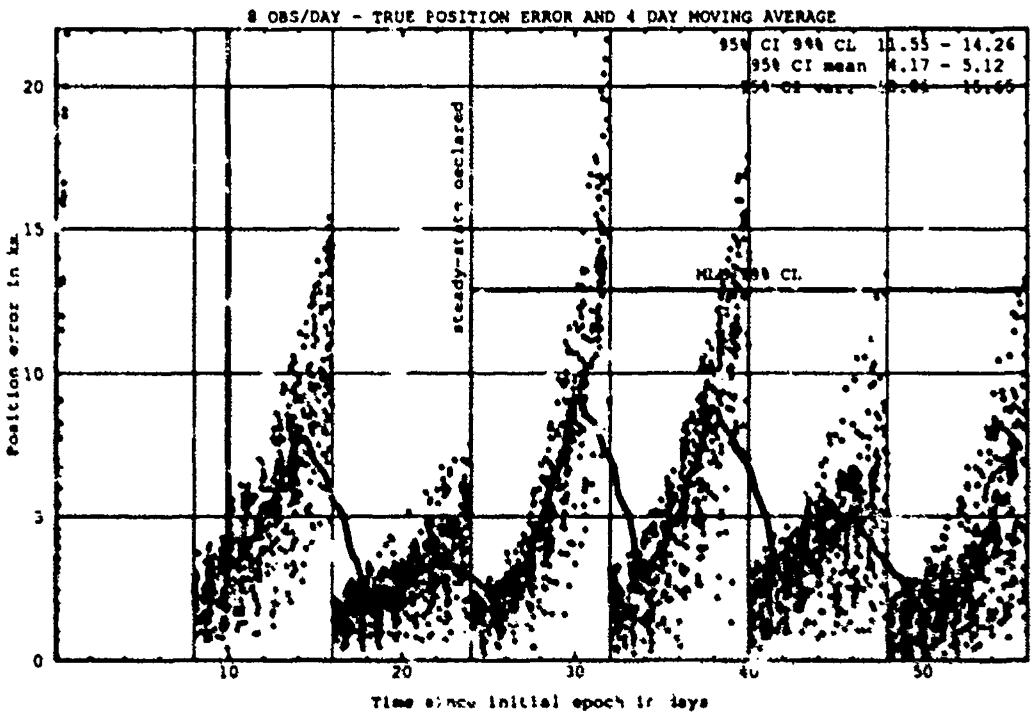


Figure I.17. Class: 3-1-1 (Catalog Number 01996), LUPI 8, OPD 6 and 8.

I.2.2 Differential Corrector Performance. **The table below provides statistical information on the last-pass residuals of the tuned differential corrector.**

Table I.4. Selected Last-Pass 95% Confidence Interval Statistics.

Catalog Number	LUPI	OPD	Mean		Variance		99% CL	
			Min	Max	Min	Max	Min	Max
	days	obs/day	km	km	km^2	km^2	km	km
01996	8	8	0.87	0.91	0.23	0.25	1.98	2.06

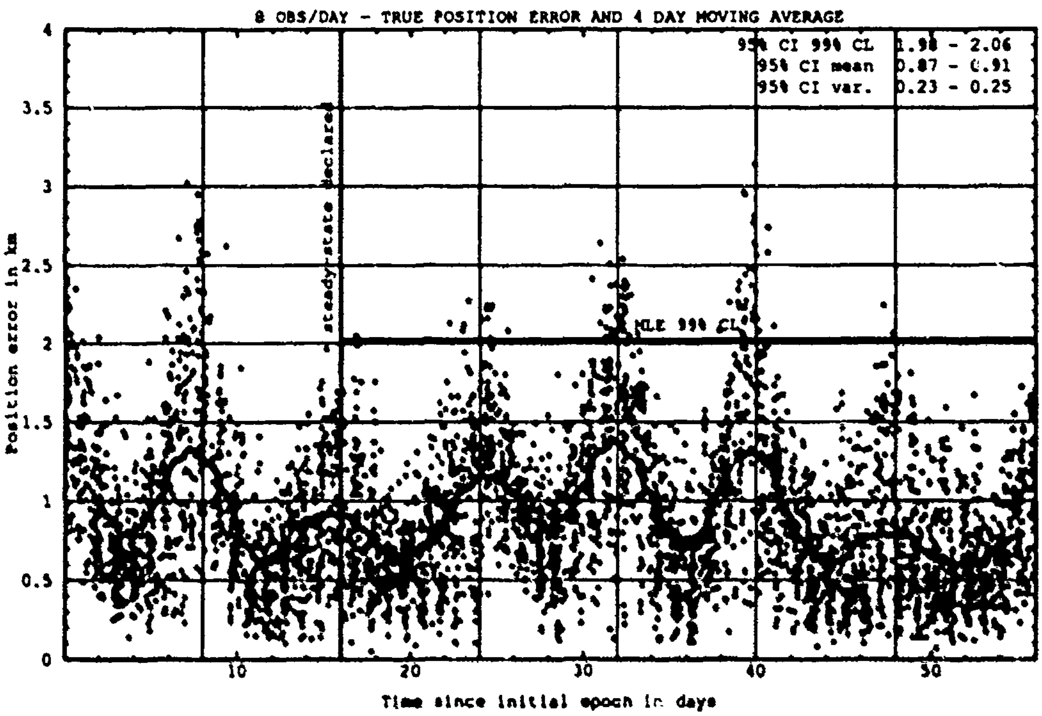


Figure I.18. Last-Pass — Class: 3-1-1 (Catalog Number 01996), LUPI 8, OPD 8.

Bibliography

1. Barton, David K. *Radar System Analysis.* Englewood Cliffs, New Jersey: Prentice-Hall, Inc. 1964.

2. Bishop, Capt Gregory, USAF, Orbital Analyst. Personal Interview. 1st Command and Control Squadron (AFSPACECOM), Cheyenne Mountain AFB CO, 22 September 1992.

3. Bishop, Capt Gregory, USAF, Orbital Analyst. Telephone Interviews. 1st Command and Control Squadron (AFSPACECOM), Cheyenne Mountain AFB CO, 24 August through 25 October 1992.

4. Casey, W.L. and D.D. Phinney. "Representative Pointed Optics and Associated Gimbal Characteristics," *Proceedings of the SPIE Acquisition, Tracking and Pointing II.* 116-123. Bellingham, WA: SPIE Press, 1988.

5. Ditto, D.H. and R.D. Huczko. "Stabilization of Large Mounts Through Auxiliary Inertial Instruments," *Proceedings of the SPIE Acquisition, Tracking and Pointing II.* 154-159. Bellingham, WA: SPIE Press, 1988.

6. Downey, George and Larry Stockum. "Electro-Optical Tracking System Requirements," *Proceedings of the SPIE Acquisition, Tracking and Pointing III.* 70-84. Bellingham, WA: SPIE Press, 1989.

7. Department of the Air Force. *Space Handbook.* AU-18. Maxwell AFB, Alabama: Air University Press, January 1985.

8. Farrago, Capt Zoltan, USAF, Astronautical Engineer. Personal Interview. United States Space Command, J3SOT, Cheyenne Mountain AFB CO, 22 September 1992.

9. "Final Report of the Space Surveillance/Command and Control (C2) Evaluation Study, Volume I, Executive Summary" (U). Contract FO5603-86-C-0002. Science Applications International Corp., Colorado Springs, CO, 24 May 1989. (AD-C045706).

10. "Final Report of the Space Surveillance/Command and Control (C2) Evaluation Study, Volume II, Surveillance Baseline" (U). Contract FO5603-86-C-0002. Science Applications International Corp., Colorado Springs, CO, 24 May 1989. (AD-C045707).

11. "Final Report of the Space Surveillance/Command and Control (C2) Evaluation Study, Volume III, Technical Data" (U). Contract FO5603-86-C-0002. Science Applications International Corp., Colorado Springs, CO, 24 May 1989. (AD-C045708).

12. First Command & Control Squadron. Operations Center Job Aid 17, Near Earth Sensor Tasking. Cheyenne Mountain AFB CO, 10 August 1992.

13. First Command & Control Squadron. Operations Center Job Aid 17, Near Earth Sensor Tasking. Cheyenne Mountain AFB CO, 4 June 1992.

14. Grissom, Wayne and others. *SATRAK Satellite Tracking Program Operator's Guide, Version 3.0.* Contract 0200121217, 0200121892. Colorado Springs, CO: Teledyne Brown Engineering, 31 May 1989.

15. High-Precision Orbit Propagator User's Manual. Microcosm Space Mission Engineering, Inc., Torrance CA, 1992.

16. Hoots, Felix R. and Roehrich, Ronald L. "Spacetrack Report 3, Models for Propagation of NORAD Element Sets." Alexandria Virginia: Defense Documentation Center. December 1980.

17. Howell, Capt J. Andreas. "The Challenge of Space Surveillance," *Sky and Telescope*: 584–588 (June 1987).

18. Jackson, Maj Paul, USAF. "Space Surveillance Satellite Catalog Maintenance," *AIAA/NASA/DOD Orbital Debris Conference: Technical Issues & Future Directions*. Paper No. 90–1339. Baltimore: American Institute of Aeronautics and Astronautics, April 1990.

19. Kelso, Lt Col Thomas S., USAF. NORAD SGP4/SDP4 Turbo Pascal Library Units, Version 2.50. School of Engineering, Air Force Institute Of Technology (AU), Wright-Patterson AFB OH, 1 October 1992.

20. Lapin, Lawrence. *Quantitative Methods for Business Decisions With Cases*. Orlando, Florida: Harcourt Brace Jovanovich, 1988.

21. Law, Averill M. and W. David Kelton. *Simulation Modeling and Analysis*. New York, New York: McGraw-Hill, Inc. 1991.

22. Love, Capt Scot, USAF. "Updates to Eglin SSN Sensor" *Space Trace*. Peterson AFB Colorado: Air Force Space Command, August 1992.

23. Morse, Maj Ronald L. and others. *Space Operations Orientation Course Handbook* (Second Edition). Colorado Springs, Colorado: FlightSafety Services Corp., 1991.

24. North American Aerospace Defense Command. *Mathematical Foundation for Space Computational Center Astrodynamic Theory*. Technical Publication SSC 008. Colorado Springs, Colorado: HQ AFSPACECOM, 8 October 1991.

25. North American Aerospace Defense Command. *Space Surveillance Center Computer Program Product Specification Abbreviations, Acronyms, and Glossary of Terms*. Technical Publication SSC 002. Colorado Springs, Colorado: HQ AFSPACECOM, 2 January 1991.

26. O'Brien, Capt Daniel L., USAF. *Preliminary Design of a Model to Assess the Effect of Space Surveillance Network (SSN) Sensor Upgrades on Orbit Prediction Accuracies Relative to the U. S. Anti-Satellite (ASAT) Mission*. MS Thesis, AFIT/GSO/ENS/ENY/91D-14. School of Engineering, Air Force Institute Of Technology (AU), Wright-Patterson AFB OH, December 1991.

27. Press, William H. and others. *Numerical Recepies in Pascal: The Art of Scientific Computing*. New York: Cambridge University Press, 1989.

28. Pritsker. A. Alan B., *Introduction to Simulation and SLAM II*. West Lafayette, Indiana: Systems Publishing Corp. 1986.

29. Rohan, P. *Surveillance Radar Performance Prediction*. London, United Kingdom: Peter Peregrinus, Ltd. 1983.

30. Schaffer, CPT Brad, USA, Astronautical Engineer. Personal Interview. United States Space Command, J3SOT, Cheyenne Mountain AFB CO, 22 September 1992.

31. Headquarters Space Division. "Space Surveillance Program, Vol I, Architecture Overview" (U). Report Number 5D-82-29. Los Angeles CA: HQ Space Division, 31 August 1981.

32. Ulich, B.L. "Overview of Acquisition, Tracking, and Pointing Technologies," *Proceedings of the SPIE Acquisition, Tracking and Pointing II.* 40-63. Bellingham, WA: SPIE Press, 1988.

33. United States Space Command. *Satellite Catalog*, Cheyenne Mountain AFB, Colorado: USSPACECOM, July 1992.

34. United States Sace Command. "Sensor Calibration Daily Weights and Biases Calibration Results." Electronic Message. 220445Z September 1992.

35. United States Space Command. *Space Surveillance Center Orbital Analysis Officer Positional Handbook* (Version 1.5C). Colorado Springs, Colorado: HQ AFSPACECOM, March 1989.

36. United States Space Command. *SSN Tactical Requirements (U).* USSPACECOM Regulation 55-12. Cheyenne Mountain AFB, Colorado: HQ USSPACECOM, 15 March 1989.

37. Wiesel, William E. Class notes for MECH 636, Advanced Astodynamics, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, April 1992.

38. Wiesel, William E. Class notes for MECH 731, Modern Methods of Orbit Determination, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, July 1992.

Vita

Captain Jeff M. Berger was born on 27 January 1963 in Oslo, Norway. As the only child of an Air Force enlisted family he lived in many places prior to his father's retiring and settling in Nazareth, Pennsylvania in 1975. Jeff graduated from Nazareth Area Senior High School in 1981 and attended Pennsylvania State University. In August 1984 he enlisted in the Air Force under the College Senior Engineering Program with the understanding that an Officer Candidate slot would be reserved for him at Officer Training School after graduation. In May 1985, then Airman 1st Class Berger graduated from Penn State with a Bachelor of Science degree in Aerospace Engineering. Following graduation he attended OTS, receiving his commission on 13 September 1985. As a new second lieutenant, his first assignment was to Air Force Systems Command, Air Force Satellite Control Network, Operating Location AB, Peterson AFB, Colorado, where he received training as a Global Positioning System Planner/Analyst. When training was completed he was reassigned to Air Force Space Command, 2nd Space Wing, as part of the initial activation group for the 1st Satellite Control Squadron at Falcon AFB, Colorado. While assigned to the 1st Satellite Control Squadron, he participated in the successful launches of the first 10 Block II Global Positioning System Satellites, was certified in the following crew positions; Planner/Analyst, Satellite Vehicle Engineer, and Flight Commander, and as a staff operations training instructor he was responsible for managing the qualification training program for all crew positions. Two promotions and an augmentation to a regular commission later, Captain Berger departed Colorado in March 1991. While enroute to his next assignment he attended Squadron Officer School in residence, afterward entering the Graduate Space Operations Program in the School of Engineering, Air Force Institute of Technology in May 1991.

Permanent address: 514 East Walnut St.
Nazareth, PA 18064

Vita

Captain Joseph B. Moles was born on 25 September 1959 in Atlanta, Georgia. He graduated from South Cobb High School in Austell, Georgia in 1977. After one year at the Georgia Institute of Technology, Captain Moles was accepted for admission into the United States Military Academy. After graduation in May 1982, Captain Moles was commissioned as a second lieutenant in the Ordnance Corps and entered the Missile and Munitions Officer Basic Course (OBC) at Redstone Arsenal, Alabama. After completing OBC, he served with two units located at Fort Bliss, Texas. In 1985 he was selected to command the 118th Ordnance Detachment, 11th Air Defense Artillery Brigade. After successfully completing command and the Ordnance Officer Advanced Course (OAC), Captain Moles was assigned to the office of the Assistant Deputy Chief of Staff for Logistics (ADCSLOG), US Army, Europe (USAREUR), as the Chief of the Operations Branch, Missile System Division and subsequently as the Chief of the Operations and Plans Branch, Munitions Division. As an additional duty, Captain Moles acted as the Inspector General for the 200th Theater Army Material Management Command from 1988 to 1990. While in Germany, Captain Moles was assigned a functional area specialty of Research and Development and selected for assignment to the Army Acquisition Corps. Additionally, Captain Moles completed the requirements for a Masters Degree in Systems Management from the University of Southern California while studying during off duty hours. After completing his tour in Germany, Captain Moles was selected for the Army's Training With Industry (TWI) program and assigned to the Northrop Corporation, Hawthorne, California, to study low observable technology. While in the TWI program Captain Moles was selected,in January of 1991, to attend the Air Force Institute Of Technology to pursue a Masters Degree in Space Operations. Captain Moles entered the School of Engineering in May 1991. Captain Moles is married but has no children except two large, pampered cats.

Permanent address: 4840 Brookwood Dr.
Mableton, GA 30059

Vita

Captain David G. Wilsey was born on 17 October 1962 in Atlantic City, New Jersey. He graduated from Mainland Regional High School in Linwood, New Jersey in 1981 and entered the University of Michigan. In 1982, he enrolled in the Air Force ROTC program and in December 1985 graduated "cum laude" with a Bachelor of Science degree in Aerospace Engineering. Upon graduation, he received his commission as a Second Lieutenant and attended Space Operations Training at the 3400th Technical Training Wing, Lowry AFB, Colorado. After completing training, he was assigned to the Defense Satellite Communication Systems Program Office, Headquarters Space Division, Los Angeles AFB, California. While there he served first as a Program Control Officer, and then as the program's Budget Execution Officer, responsible for over 500 million dollars in active appropriations. In March 1988, he was reassigned to the 16th Surveillance Squadron, Shemya AFB, Alaska, as a Space Surveillance Operations Crew Commander, commanding a 25 person operations and support crew responsible for the operations of the Cobra Dane radar system in performance of intelligence collection, ballistic missile early warning, and space surveillance missions. He also served as a line crew instructor and evaluator, Squadron Intelligence Officer, and Chief of Standardization and Evaluation. In May 1989, he departed Alaska, attended the 1013th Combat Crew Training Squadron, and reported to the 5th Defense Space Communications Squadron, Woomera, South Australia, as a Satellite Operations Crew Commander. There he commanded a 48 person multi-service/multinational operations and support crew responsible for the operation of DSP satellite and ground station resources in the performance of ballistic missile early warning and space launch reporting missions. He also served as a line crew instructor, and the Squadron Intelligence Officer. Departing Australia in March 1991, he attended Squadron Officer School in residence and entered the School of Engineering, Air Force Institute of Technology in May 1991.

Permanent address: 22 Georgetown Ct.
Linwood, NJ 08221

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1992	Master's Thesis

4. TITLE AND SUBTITLE
AN ANALYSIS OF USSPACECOM'S SPACE SURVEILLANCE NETWORK SENSOR TASKING METHODOLOGY

5. FUNDING NUMBERS

6. AUTHOR(S)
Jeff M. Berger, Captain, USAF
Joseph B. Moles, Captain, USA
David G. Wilsey, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GSO/ENG/92D-02

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
1CACS/CC
1 Norad Rd. Suite 3105
CMAFB CO 80916

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release:
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
This study provides the basis for the development of a cost/benefit assessment model to determine the effects of alterations to the Space Surveillance Network (SSN) on orbital element (OE) set accuracy. It provides a review of current methods used by NORAD and the SSN to gather and process observations, an alternative to the current Gabbard classification method, and the development of a model to determine the effects of observation rate and correction interval on OE set accuracy. The proposed classification scheme is based on satellite J_2 perturbations. Specifically, classes were established based on mean motion, eccentricity, and inclination since J_2 perturbation effects are functions of only these elements. Model development began by creating representative sensor observations using a highly accurate orbital propagation model. These observations were compared to predicted observations generated using the NORAD Simplified General Perturbation (SGP4) model and differentially corrected using a Bayes, sequential estimation, algorithm. A 10-run Monte Carlo analysis was performed using this model on 12 satellites using 16 different observation rate/correction interval combinations. An ANOVA and confidence interval analysis of the results show that this model does demonstrate the differences in steady state position error based on varying observation rate and correction interval.

14. SUBJECT TERMS
Space Surveillance Network, Space Surveillance Center, Orbit Determination, Differential Correction, Gabbard Classes, Sensor Tasking

15. NUMBER OF PAGES
407

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C - Contract		**PR** - Project	
G - Grant		**TA** - Task	
PE - Program Element		**WU** - Work Unit Accession No.	

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with....; Trans. of....; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE - See authorities.
NASA - See Handbook NHB 2200.2.
NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - Leave blank.
DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA - Leave blank.
NTIS - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.